APPENDIX B

MATLAB Source Code

```
function data = Ambient_Correction( data )
% This function applies correction factors to the measured data according
% to section S6.7 of the final rule.

%-------------------------------------------------------------------------
% Correct Overall Levels
%-------------------------------------------------------------------------
% See Table 8 of the final rule

for ink = 1 : length( data.vehicleOverallLevels_dBA )
        delta = data.vehicleOverallLevels_dBA ( ink ) - data.ambientOverallLevels_dBA ( ink ) ;
        if delta > 10
            data.vehicleOverallLevels_dBA ( ink ) = data.vehicleOverallLevels_dBA ( ink );
        elseif delta > 8 && delta <= 10
            data.vehicleOverallLevels_dBA ( ink ) = data.vehicleOverallLevels_dBA ( ink )- 0.5;
        elseif delta > 6 && delta <= 8
            data.vehicleOverallLevels_dBA ( ink ) = data.vehicleOverallLevels_dBA ( ink ) - 1.0;
        elseif delta > 4.5 && delta <= 6
            data.vehicleOverallLevels_dBA ( ink ) = data.vehicleOverallLevels_dBA ( ink ) - 1.5;
        elseif delta > 3 && delta <= 4.5
            data.vehicleOverallLevels_dBA ( ink ) = data.vehicleOverallLevels_dBA ( ink ) - 2.5;
        elseif delta <= 3
            data.vehicleOverallLevels_dBA ( ink ) = 0;
        end
end

%-------------------------------------------------------------------------
% Correct One-third Octave Bands
%-------------------------------------------------------------------------
% See Table 9 of the final rule

for ink = 1 : size( data.vehicleFOBLevels_dBA, 1 )
    for lead = 1 : size( data.vehicleFOBLevels_dBA, 2 )
        delta = data.vehicleFOBLevels_dBA( ink, lead ) - data.ambientFOBLevels_dBA( ink, lead );
        if delta > 6
            data.vehicleFOBLevels_dBA( ink, lead ) = data.vehicleFOBLevels_dBA( ink, lead );
        elseif delta > 4.5 && delta <= 6
            data.vehicleFOBLevels_dBA( ink, lead ) = data.vehicleFOBLevels_dBA( ink, lead ) - 1.5;
        elseif delta > 3 && delta <= 4.5
            data.vehicleFOBLevels_dBA( ink, lead ) = data.vehicleFOBLevels_dBA( ink, lead ) - 2.5;
        elseif delta <= 3
            data.vehicleFOBLevels_dBA( ink, lead ) = 0;
        end
    end
end
```

```
function idx = Check_2dB( driverOverall_dBA, passengerOverall_dBA, handles, frontOverall_dBA )
% S7.1.3: Identify first four valid tests within 2dBA.
%
% (a) For each valid test sound file identified in S7.1.2, determine a maximum
% overall SPL value, in decibels. Each SPL value will be reported to the
% nearest tenth of a decibel.
%
% (b) Compare the first four left-side SPL values from S7.1.3(a) of this
% paragraph, and determine the range by taking the difference between the
% largest and smallest of the four values. In the same manner, determine the
% range of SPL values for the first four right-side and the first four front-
% center sound files. ****** If the range for the left side, right side, and
% front-center are all less than or equal to 2.0 dB, then ****** the twelve
% sound files associated with the first four valid tests will be used for the
% one-third octave band evaluations in S7.1.5. and S7.1.6. If the range of
% the SPL values for the left side are not within 2 dBA, or for the right
% side are not within 2 dBA, or for the front-center of the vehicle are not
% within 2 dBA, an iterative process will be used to consider sound files
% from additional sequential tests until the range for all three microphone
% locations are within 2 dBA for the same sequence number recordings for all
% three locations

% S7.2 Stationary vehicle in reverse gear.  Test the vehicle per S7.1.1
% through S7.1.5 except that the rear plane of the vehicle is placed on the
% PP' line, no center microphone is used, and the vehicle's transmission gear
% selector is placed in the 'Reverse' position.  The minimum sound level
% requirements for the Reverse test condition are contained in S5.1.2,
% Table 2, for 4-band compliance and in S5.2, Table 6, for 2-band compliance.

sampleSize = 4; % Start by looking at groups of four (may increase within the loop)
while sampleSize > 3
    if sampleSize > length ( driverOverall_dBA )
       set( handles.display2dBError, 'Visible', 'on' );
       set( handles.display2dBError, 'String', 'No combination of 4 samples are within 2dB' );
       set( handles.display2dBError, 'ForegroundColor', [1, 0, 0 ] );
       set( handles.pushbuttonCompute, 'Enable', 'on' );
       break

    end
    idxNumberSpace = 1 : sampleSize; % Gives list of indeces to be considered:
    %                                  1-4 gives only one option for idx (1 2 3 4 )
    %                                  1-5 = options 5 options ( 1 2 3 4 )
    %                                                          ( 1 2 3 5 )
    %                                                          ( 1 2 4 5 )
    %                                                          ( 1 3 4 5 )
    %                                                          ( 2 3 4 5 ), etc
    idx = nchoosek( idxNumberSpace, 4 ); % Each row gives one possible combination of 4 samples from the (equal or)
    larger sampleSize

    % Check Driver Side
```

```
    for ink = 1 : size( idx, 1 )
        checkDriver( ink ) = max( driverOverall_dBA( idx( ink, : ) ) ) - min( driverOverall_dBA( idx( ink, : ) ) );
        %this is using 1 decimal precision, don't need to round to 1 decimal again
    end
    boolSort = checkDriver <= 2;
    idx = idx( checkDriver <= 2, : );
    if isempty( idx )
        sampleSize = sampleSize + 1;

    else % Driver OK, Check Passenger Side
        for ink = 1 : size( idx, 1 ) % Just compare the ones that are within 2 dB for driver side
            checkPassenger( ink ) = max( passengerOverall_dBA( idx( ink, : ) ) ) - min( passengerOverall_dBA( idx( ink
            , : ) ) );
        end
        idx = idx( checkPassenger <= 2, : ); % This now includes the idxs that meet requirement for both sides
        if isempty( idx )
            sampleSize = sampleSize + 1;

        elseif exist( 'frontOverall_dBA', 'var' ) % Passenger OK, Check Front Side
            for ink = 1 : size( idx, 1 ) % Just compare the ones that are within 2 dB for driver and passenger side
                checkFront( ink ) = max( frontOverall_dBA( idx( ink, : ) ) ) - min( frontOverall_dBA( idx( ink, : ) )
                );
            end
            idx = idx( checkFront <= 2, : ); % This now includes the idxs that meet requirement for all sides
            if isempty( idx )
                sampleSize = sampleSize + 1;

            else % Front Side OK, idx meet tolerance for all sides
                % If idx is just one row, then unambiguous, if more than one
                % need to take the one with the lowest idx sum (or the first one
                % with the lowest idx sum)
                idxSum = sum( idx, 2 );
                [ ~, idxMin ] = min( idxSum );
                idx = idx( min( idxMin ), : );
                sampleSize = 0;
            end

        else % Passenger OK, No Front Side, idx meet tolerance for all sides
            % If idx is just one row, then unambiguous, if more than one
            % need to take the one with the lowest idx sum (or the first one
            % with the lowest idx sum)
            idxSum = sum( idx, 2 );
            [ ~, idxMin ] = min( idxSum );
            idx = idx( min( idxMin ), : );
            sampleSize = 0;
        end
    end
end


```

```
function handles = ComplianceAnalysis_GUIPreprocessing( handles )
% This code prepares data for compliance testing according to the final rule

%% Input Data

%Op Conditions
% 1 - Reverse
% 2 - Stationary
% 3 - 10 km/h
% 4 - 20 km/h
% 5 - 30 km/hfor ink = 1 : 5  % looping through op types

% For switches
loadPrecomputedData = 0 ;
reporting = 1 ;

NHTSAsampleData = cell( 1, 5 );
allianceSampleData = cell( 1, 5 );
frontSideThirdOctaveAvg_dBA = -88 * ones( 1, 13 );
oBs = {'315 Hz', '400 Hz', '500Hz', '630 Hz', '800 Hz', '1000 Hz', '1250 Hz', '1600 Hz', '2000 Hz', '2500 Hz', '3150
Hz', '4000 Hz', '5000 Hz'} ;

opCondOptions = ({ 'Reverse', 'Stationary', '10 km/h', '20 km/h', '30 km/h' }) ;
opCondOptionsSimple = ({ 'Reverse', 'Stationary', '10kmh', '20kmh', '30kmh' }) ;

%**************************************************************************
%**************************************************************************
% Handles conversion to variables
Make =  char( get( handles.editMake, 'String' ) );
Model = char( get( handles.editModel, 'String' ) );
year = char( get( handles.editYear, 'String' ) );
%**************************************************************************
%**************************************************************************


%% File Numbering for Reporing

fileNameRoot = sprintf( '%s%s%s', year, Make, Model ) ;

try
filesInDirAll = dir ( handles.outputDirectory );
catch
    handles.outputFileCancel = true ;
    return
end

filesInDir = filesInDirAll( ~ismember( { filesInDirAll.name }, { '.','..' } ) );
numFiles =  sum( ~cellfun( 'isempty', strfind( { filesInDir.name }, fileNameRoot ) ) );

%Finding number of files for each operating condition in the output
```

```
%directory, used for report file numbers
if numFiles >= 1
    fileNameLengthReverse = length( sprintf('%s_Reverse', fileNameRoot ) ) ;
    fileNameLengthStationary = length( sprintf('%s_Stationary', fileNameRoot ) );
    fileNameLengthPassby = length( sprintf('%s_10kmh', fileNameRoot ) ) ;

    %Finding the files that correspond to Reverse, Stationary, or Passby
    %operating conditions
    filesInDirReverseIdx = ~cellfun( 'isempty', strfind( { filesInDir.name }, 'Reverse' ) ) ;
    filesInDirStationaryIdx = ~cellfun( 'isempty', strfind( { filesInDir.name }, 'Stationary' ) ) ;
    filesInDirPassbyIdx1 = strfind( { filesInDir.name }, 'kmh' )  ;
    filesInDirPassbyIdx2 = find( not( cellfun('isempty', filesInDirPassbyIdx1 ) ) ) ;

    filesInDirReverse = filesInDir( filesInDirReverseIdx ) ;
    filesInDirStationary = filesInDir( filesInDirStationaryIdx ) ;
    filesInDirPassby = filesInDir( filesInDirPassbyIdx2 ) ;

    numberInc = 0;

    % Finding the highest number report among all operating conditions from
    % all previous runs that have been reported out to this output directory so that
    % the group of reports from this analysis will have a consistent report
    % number and it will be the highest number in this output directory.
    for ink = 1: length( filesInDirReverse )
        fileNumbersReverse( ink ) =  str2double( filesInDirReverse( ink ).name( fileNameLengthReverse + 2 : end - 5 )
        ) ;
        numberInc = max( [ numberInc max( fileNumbersReverse ) ] );
    end

    for ink = 1: length( filesInDirStationary )
        fileNumbersStationary( ink ) =  str2double( filesInDirStationary( ink ).name( fileNameLengthStationary + 2 :
        end - 5 ) ) ;
        numberInc = max( [ numberInc max( fileNumbersStationary ) ] );
    end

    for ink = 1: length( filesInDirPassby )
        fileNumbersPassby( ink ) =  str2double( filesInDirPassby( ink ).name( fileNameLengthPassby + 2 : end - 5 ) ) ;
        numberInc = max( [ numberInc max( fileNumbersStationary ) ] );
    end

    numberInc = numberInc + 1 ;

else

    numberInc = 1 ;

end

handles.numberInc = numberInc ;

```

```
%% Data Prep

for ink = 1 : 5

    % ------------------------------------------------------------------
    % For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
    % ------------------------------------------------------------------

    handles.opCondForLoop = opCondOptions{ ink } ;

    %% Select signal data and process through SLM

    %-----------------------------------------------------------------------------------------------
    %Final Rule: S7.3.3(a)
    %For each valid test sound file identified in S7.3.2, determine a maximum overall
    %SPL value, in decibels. The SPL value will be reported to the nearest tenth of a decibel.
    %-----------------------------------------------------------------------------------------------

    tic
    handles = Compute_SPL_by_Vehicle_Side_forGUI( handles );
    drawnow
    if handles.stopButtonFlag
        % For stop button
        return;
    end

    %**********************************************************************
    %**********************************************************************
    % handles conversion to variables
    resultsDriver = handles.driverSPL ;
    resultsPassenger = handles.passengerSPL ;
    resultsFront = handles.frontSPL ;
    %**********************************************************************
    %**********************************************************************

    handles.opCondSimple = opCondOptionsSimple{ ink } ;

    eval( [ sprintf( 'resultsDriver_%s', handles.opCondSimple ) ' = resultsDriver;' ] )
    eval( [ sprintf( 'resultsPassenger_%s', handles.opCondSimple ) ' = resultsPassenger;' ] )
    eval( [ sprintf( 'resultsFront_%s', handles.opCondSimple ) ' = resultsFront;' ] )

    driverInputFileName = resultsDriver.inputFileNames ;
    driverAmbientFileName = resultsDriver.ambientFileNames ;
    driverAmbientSide = resultsDriver.ambientLowerSide ;
    passengerInputFileName = resultsPassenger.inputFileNames ;
```

```
148       passengerAmbientFileName = resultsPassenger.ambientFileNames ;
149       passengerAmbientSide = resultsPassenger.ambientLowerSide ;
150
151       if ink == 2 ; %for stationary only
152
153           frontInputFile = resultsFront.inputFileNames ;
154           frontAmbientInputFile = resultsFront.ambientFileNames ;
155           fileNames.side{ ink } = resultsFront.inputFileNames ;
156           fileNames.sideambient{ ink } = resultsFront.ambientFileNames ;
157
158       end
159
160       try
161           save( 'timeSeriesResults.mat', sprintf( 'resultsDriver_%s', handles.opCondSimple ), sprintf(
              'resultsPassenger_%s', handles.opCondSimple ), sprintf( 'resultsFront_%s', handles.opCondSimple ), '-append' );
162       catch
163           save( 'timeSeriesResults.mat', sprintf( 'resultsDriver_%s', handles.opCondSimple ), sprintf(
              'resultsPassenger_%s', handles.opCondSimple ), sprintf( 'resultsFront_%s', handles.opCondSimple ) );
164           toc
165
166       end
167
168       %% Choose four samples within 2 dB of each other:
169       % -----------------------------------------------------------------
170       % For stop button
171       drawnow
172       if get(handles.pushbuttonStop, 'userdata') % stop condition
173           handles.stopButtonFlag = true ;
174           return;
175       end
176       % -----------------------------------------------------------------
177
178       %-------------------------------------------------------------------------------------------------
179       %Final Rule: S7.3.3(b)
180       % Compare the first four left side maximum overall SPL values. Of the four SPL values
181       % calculate the difference between the largest and smallest maximum SPL values. The same
182       % process will be used to determine the difference between the largest and smallest maximum
183       % SPL values for the first four right side maximum SPL values. If the difference values on
184       % the left and right sides of the test vehicle are both less than or equal to 2.0 dB, then
185       % the eight sound files associated with the first four valid tests will be used for the final
186       % one-third octave band evaluation in accordance with S7.3.4. and S7.3.5. If the first four
187       % test sound files on each side of the vehicle are not within 2 dBA, an iterative process will
188       % be used to consider sound files from additional sequential tests until the range for both
189       % microphone locations are within 2 dBA for the same sequence number recordings for both locations.
190       %-------------------------------------------------------------------------------------------------
191       disp( 'Choosing 4 within 2 dB.' )
192
193       set( handles.textProcessingStep, 'String', 'Choosing 4 samples within 2 dB' );
194       set( handles.displayRunStatusCurrentFile, 'Visible', 'off' );
195       set( handles.textRunStatusOf, 'Visible', 'off');
```

```
set( handles.displayRunStatusTotalFiles, 'Visible', 'off' );
drawnow

if strcmpi( handles.opCondSimple, 'Stationary' ) % Only do front if Stationary and front was computed
    idx  = Check_2dB( resultsDriver.vehicleOverallLevels_dBA, ...
        resultsPassenger.vehicleOverallLevels_dBA, handles, ...
        resultsFront.vehicleOverallLevels_dBA );

    % Trim down to four samples
    resultsFront.vehicleOverallLevels_dBA = resultsFront.vehicleOverallLevels_dBA( idx );
    resultsFront.vehicleFOBLevels_dBA = resultsFront.vehicleFOBLevels_dBA( idx, : );
    resultsFront.ambientOverallLevels_dBA = resultsFront.ambientOverallLevels_dBA( idx );
    resultsFront.ambientFOBLevels_dBA = resultsFront.ambientFOBLevels_dBA( idx, : );

else
    idx = Check_2dB( resultsDriver.vehicleOverallLevels_dBA, ...
        resultsPassenger.vehicleOverallLevels_dBA, handles );
end

% Trim down to four samples
resultsDriver.vehicleOverallLevels_dBA = resultsDriver.vehicleOverallLevels_dBA( idx );
resultsDriver.vehicleFOBLevels_dBA = resultsDriver.vehicleFOBLevels_dBA( idx, : );
resultsDriver.ambientOverallLevels_dBA = resultsDriver.ambientOverallLevels_dBA( idx );
resultsDriver.ambientFOBLevels_dBA = resultsDriver.ambientFOBLevels_dBA( idx, : );

resultsPassenger.vehicleOverallLevels_dBA = resultsPassenger.vehicleOverallLevels_dBA( idx );
resultsPassenger.vehicleFOBLevels_dBA = resultsPassenger.vehicleFOBLevels_dBA( idx, : );
resultsPassenger.ambientOverallLevels_dBA = resultsPassenger.ambientOverallLevels_dBA( idx );
resultsPassenger.ambientFOBLevels_dBA = resultsPassenger.ambientFOBLevels_dBA( idx, : );

%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
% Write intermediate results to .xlsx

warning( 'off' ,  'MATLAB:openvar:VariableNameContainsSpaces' );
warning('off','MATLAB:xlswrite:AddSheet');

if reporting == 1 ;

    % File naming
    fileNameWithOpCond = sprintf( '%s_%s', fileNameRoot, handles.opCondSimple ) ;
    fileName = sprintf( '%s_%d.xlsx', fileNameWithOpCond, numberInc ) ;
    fileNamePath = fullfile( handles.outputDirectory, fileName) ;

    % Headers
    sheetSamplesHeader = strcat('Four samples used in analysis for', {' '}, year, {' '}, Make, {' '}, Model, ',',
    {' '}, handles.opCondSimple, ' operating condition') ;
    overallHeader = {'Overall Results - 4 Samples'} ;
    overallHeaderDriverVehicle = {'Overall Driver Vehicle Results'} ;
    overallHeaderDriverAmbient = {'Overall Driver Ambient Results'} ;
    overallHeaderPassengerVehicle = {'Overall Passenger Vehicle Results'} ;
```

```
        overallHeaderPassengerAmbient = {'Overall Passenger Ambient Results'} ;
        FOBHeader = {'FOB Results - 4 Samples'} ;
        FOBHeaderDriverVehicle = {'FOB Driver Vehicle Results'} ;
        FOBHeaderDriverAmbient = {'FOB Driver Ambient Results'} ;
        FOBHeaderPassengerVehicle = {'FOB Passenger Vehicle Results'} ;
        FOBHeaderPassengerAmbient = {'FOB Passenger Ambient Results'} ;

        % Results
        xlswrite (fileNamePath,overallHeader, '4Samples', 'A3')
        xlswrite (fileNamePath,overallHeaderDriverVehicle, '4Samples', 'A4')
        xlswrite(fileNamePath, resultsDriver.vehicleOverallLevels_dBA, '4Samples', 'A5')
        xlswrite (fileNamePath,overallHeaderDriverAmbient, '4Samples', 'A6')
        xlswrite(fileNamePath, resultsDriver.ambientOverallLevels_dBA, '4Samples', 'A7')
        xlswrite (fileNamePath,overallHeaderPassengerVehicle, '4Samples', 'A8')
        xlswrite(fileNamePath, resultsPassenger.vehicleOverallLevels_dBA, '4Samples', 'A9')
        xlswrite (fileNamePath,overallHeaderPassengerAmbient, '4Samples', 'A10')
        xlswrite(fileNamePath, resultsPassenger.ambientOverallLevels_dBA, '4Samples', 'A11')

        xlswrite (fileNamePath, FOBHeaderDriverVehicle, '4Samples', 'A14')
        xlswrite (fileNamePath, oBs, '4Samples', 'A15')
        xlswrite (fileNamePath, resultsDriver.vehicleFOBLevels_dBA, '4Samples', 'A16')
        xlswrite (fileNamePath, FOBHeader, '4Samples', 'A13')

        xlswrite (fileNamePath, FOBHeaderDriverAmbient, '4Samples', 'A21')
        xlswrite (fileNamePath, oBs, '4Samples', 'A22')
        xlswrite (fileNamePath, resultsDriver.ambientFOBLevels_dBA, '4Samples', 'A23')

        xlswrite (fileNamePath, FOBHeaderPassengerVehicle, '4Samples', 'A28')
        xlswrite (fileNamePath, oBs, '4Samples', 'A29')
        xlswrite (fileNamePath, resultsPassenger.vehicleFOBLevels_dBA, '4Samples', 'A30')

        xlswrite (fileNamePath, FOBHeaderPassengerAmbient, '4Samples', 'A35')
        xlswrite (fileNamePath, oBs, '4Samples', 'A36')
        xlswrite (fileNamePath, resultsPassenger.ambientFOBLevels_dBA, '4Samples', 'A37')

        xlswrite (fileNamePath,sheetSamplesHeader, '4Samples', 'A1')

    end
    %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

    %% Ambient Correction
    % -----------------------------------------------------------------
    % For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
    % -----------------------------------------------------------------

```

```
    %--------------------------------------------------------------------------------------------------
    % Final Rule S7.3.4(b)
    % Each of the eight acoustic sound data file maximum overall SPL values will be corrected for the
    % recorded ambient conditions as specified in paragraph S6.7. The test results will be reported to
    % the nearest tenth of a decibel.
    %--------------------------------------------------------------------------------------------------

    disp( 'Ambient Correction.' )
    set( handles.textProcessingStep, 'String', 'Correcting for ambient' );
    set( handles.displayRunStatusCurrentFile, 'Visible', 'off' );
    set( handles.textRunStatusOf, 'Visible', 'off');
    set( handles.displayRunStatusTotalFiles, 'Visible', 'off' );
    drawnow

    if strcmpi( handles.opCondSimple, 'Stationary' ) % Only do front if Stationary Front Computed
        resultsFrontCorrected = Ambient_Correction( resultsFront );
    end
    resultsDriverCorrected = Ambient_Correction( resultsDriver );
    resultsPassengerCorrected = Ambient_Correction( resultsPassenger );

    %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    % Write intermediate results to .xlsx

    if reporting == 1 ;
        % Headers
        sheetAmbiCorOverallHeader = strcat('Four samples used in analysis, corrected for ambient for', {' '}, year, {
        ' '}, Make, {' '}, Model, ',', {' '}, handles.opCondSimple, ' operating condition') ;
        ambiCorOverallHeader = {'Overall Results Corrected for Ambient - 4 Samples'} ;
        ambiCorOverallHeaderDriverVehicle = {'Overall Driver Vehicle Results - Corrected for Ambient'} ;
        ambiCorOverallHeaderDriverAmbient = {'Overall Driver Ambient Results - Corrected for Ambient'} ;
        ambiCorOverallHeaderPassengerVehicle = {'Overall Passenger Vehicle Results - Corrected for Ambient'} ;
        ambiCorOverallHeaderPassengerAmbient = {'Overall Passenger Ambient Results - Corrected for Ambient'} ;
        ambiCorFOBHeader = {'FOB Results Corrected for Ambient - 4 Samples'} ;
        ambiCorFOBHeaderDriverVehicle = {'FOB Driver Vehicle Results - Corrected for Ambient'} ;
        ambiCorFOBHeaderDriverAmbient = {'FOB Driver Ambient Results - Corrected for Ambient'} ;
        ambiCorFOBHeaderPassengerVehicle = {'FOB Passenger Vehicle Results - Corrected for Ambient'} ;
        ambiCorFOBHeaderPassengerAmbient = {'FOB Passenger Ambient Results - Corrected for Ambient'} ;

        % Results
        xlswrite (fileNamePath,ambiCorOverallHeader, '4SamplesCorrected', 'A3')
        xlswrite (fileNamePath,ambiCorOverallHeaderDriverVehicle, '4SamplesCorrected', 'A4')
        xlswrite(fileNamePath, resultsDriverCorrected.vehicleOverallLevels_dBA, '4SamplesCorrected', 'A5')
        xlswrite (fileNamePath,ambiCorOverallHeaderDriverAmbient, '4SamplesCorrected', 'A6')
        xlswrite(fileNamePath, resultsDriverCorrected.ambientOverallLevels_dBA, '4SamplesCorrected', 'A7')
        xlswrite (fileNamePath,ambiCorOverallHeaderPassengerVehicle, '4SamplesCorrected', 'A8')
        xlswrite(fileNamePath, resultsPassengerCorrected.vehicleOverallLevels_dBA, '4SamplesCorrected', 'A9')
        xlswrite (fileNamePath,ambiCorOverallHeaderPassengerAmbient, '4SamplesCorrected', 'A10')
        xlswrite(fileNamePath, resultsPassengerCorrected.ambientOverallLevels_dBA , '4SamplesCorrected', 'A11')

        xlswrite (fileNamePath, ambiCorFOBHeader, '4SamplesCorrected', 'A13')
```

```matlab
        xlswrite (fileNamePath, ambiCorFOBHeaderDriverVehicle, '4SamplesCorrected', 'A14')
        xlswrite (fileNamePath, oBs, '4SamplesCorrected', 'A15')
        xlswrite (fileNamePath, resultsDriverCorrected.vehicleFOBLevels_dBA, '4SamplesCorrected', 'A16')

        xlswrite (fileNamePath, ambiCorFOBHeaderDriverAmbient, '4SamplesCorrected', 'A21')
        xlswrite (fileNamePath, oBs, '4SamplesCorrected', 'A22')
        xlswrite (fileNamePath, resultsDriverCorrected.ambientFOBLevels_dBA, '4SamplesCorrected', 'A23')

        xlswrite (fileNamePath, ambiCorFOBHeaderPassengerVehicle, '4SamplesCorrected', 'A28')
        xlswrite (fileNamePath, oBs, '4SamplesCorrected', 'A29')
        xlswrite (fileNamePath, resultsPassengerCorrected.vehicleFOBLevels_dBA, '4SamplesCorrected', 'A30')

        xlswrite (fileNamePath, ambiCorFOBHeaderPassengerAmbient, '4SamplesCorrected', 'A35')
        xlswrite (fileNamePath, oBs, '4SamplesCorrected', 'A36')
        xlswrite (fileNamePath, resultsPassengerCorrected.ambientFOBLevels_dBA, '4SamplesCorrected', 'A37')

        xlswrite (fileNamePath,sheetAmbiCorOverallHeader, '4SamplesCorrected', 'A1')

    end
    %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

    %% Average Samples for All Sides
    % -----------------------------------------------------------------
    % For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
    % -----------------------------------------------------------------

    %--------------------------------------------------------------------------------------------------
    % Final Rule S7.3.4(c)
    % Calculate the average of the four overall ambient-corrected SPL values on each side of the vehicle
    % to derive one corrected maximum overall SPL value for each side of the vehicle. The result will be
    % reported to the nearest tenth of a decibel.

    % Final Rule S7.3.5(a)
    % The side of the vehicle selected in S7.3.4 will have four associated individual acoustic sound
    % data files. Each sound file shall be broken down into its one-third octave band levels.

    % Final Rule S7.3.5(b)
    % The identified octave band levels in each of the four sound files will be corrected for the measured
    % ambient levels as specified in paragraph S6.7.

    % Final Rule S7.3.5(c)
    % The four corrected sound pressure level values calculated from each of the four sound files in
    % each one-third octave band will be averaged together to get the average corrected sound pressure
    % level in each one-third octave band.
    %--------------------------------------------------------------------------------------------------
```

```
    disp( 'Averaging' )
    set( handles.textProcessingStep, 'String', 'Averaging' );
    set( handles.displayRunStatusCurrentFile, 'Visible', 'off' );
    set( handles.textRunStatusOf, 'Visible', 'off');
    set( handles.displayRunStatusTotalFiles, 'Visible', 'off' );
    drawnow

    if strcmpi( handles.opCondSimple, 'Stationary' ) % Only do front if Stationary Front Computed
        frontSideOverallAvg_dBA = round(10 * mean( resultsFrontCorrected.vehicleOverallLevels_dBA ) ) / 10;
        frontSideThirdOctaveAvg_dBA = round(10 * mean( resultsFrontCorrected.vehicleFOBLevels_dBA, 1 ) ) / 10;
    end

    driverSideOverallAvg_dBA = round(10 * mean( resultsDriverCorrected.vehicleOverallLevels_dBA ) ) / 10 ;
    driverSideThirdOctaveAvg_dBA = round(10 * mean( resultsDriverCorrected.vehicleFOBLevels_dBA, 1 ) ) / 10;
    passengerSideOverallAvg_dBA = round(10 * mean( resultsPassengerCorrected.vehicleOverallLevels_dBA ) ) / 10;
    passengerSideThirdOctaveAvg_dBA = round(10 * mean( resultsPassengerCorrected.vehicleFOBLevels_dBA, 1 ) ) / 10;

    %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    % Write intermediate results to .xlsx

    if reporting == 1 ;

        % Headers
        sheetAvgOverallHeader = strcat('Results Corrected for Ambient, Averaged for', {' '}, year, {' '}, Make, {' '}, Model, ',', {' '}, handles.opCondSimple, ' operating condition') ;
        AvgOverallHeader = {'Overall Results Corrected for Ambient, Averaged' } ;
        AvgOverallHeaderDriverVehicle = {'Driver Vehicle'} ;
        AvgOverallHeaderPassengerVehicle = {'Passenger Vehicle'} ;
        AvgFOBHeader = {'FOB Results Corrected for Ambient, Averaged'} ;
        AvgFOBHeaderDriverVehicle = {'Driver Vehicle'} ;
        AvgFOBHeaderPassengerVehicle = {'Passenger Vehicle'} ;

        % Results
        xlswrite (fileNamePath, AvgOverallHeader, 'Averaged', 'A3')
        xlswrite (fileNamePath, AvgOverallHeaderDriverVehicle, 'Averaged', 'A4')
        xlswrite(fileNamePath, driverSideOverallAvg_dBA, 'Averaged', 'A5')
        xlswrite (fileNamePath, AvgOverallHeaderPassengerVehicle, 'Averaged', 'A6')
        xlswrite(fileNamePath, passengerSideOverallAvg_dBA, 'Averaged', 'A7')

        xlswrite (fileNamePath,AvgFOBHeader, 'Averaged', 'A9')
        xlswrite (fileNamePath,AvgFOBHeaderDriverVehicle, 'Averaged', 'A10')
        xlswrite (fileNamePath, oBs, 'Averaged', 'A11')
        xlswrite(fileNamePath, driverSideThirdOctaveAvg_dBA, 'Averaged', 'A12')

        xlswrite (fileNamePath,AvgFOBHeaderPassengerVehicle, 'Averaged', 'A14')
        xlswrite (fileNamePath, oBs, 'Averaged', 'A15')
        xlswrite(fileNamePath, passengerSideThirdOctaveAvg_dBA, 'Averaged', 'A16')

        xlswrite (fileNamePath, sheetAvgOverallHeader, 'Averaged', 'A1')
```

```
    end
    %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

    %% Take Lower Side
    % ----------------------------------------------------------------
    % For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
    % ----------------------------------------------------------------

    %-------------------------------------------------------------------------------------------------
    % Final Rule S7.3.4(d)
    % The side of the vehicle with the lowest average corrected maximum overall SPL value will be the
    % side of the vehicle that is further evaluated for compliance at the one-third octave band levels
    % in accordance with S7.3.5.

    % S7.1.5a - look at LEFT and RIGHT mics ONLY for detection (2 and 4
    % band tests)
    %-------------------------------------------------------------------------------------------------

    disp( 'Taking Lower Side' )
    set( handles.textProcessingStep, 'String', 'Taking lower side' );
    set( handles.displayRunStatusCurrentFile, 'Visible', 'off' );
    set( handles.textRunStatusOf, 'Visible', 'off');
    set( handles.displayRunStatusTotalFiles, 'Visible', 'off' );
    drawnow

    if driverSideOverallAvg_dBA < passengerSideOverallAvg_dBA
        finalOverallLevels_dBA = driverSideOverallAvg_dBA;
        finalThirdOBlevels_dBA = driverSideThirdOctaveAvg_dBA;
        lowerSide = 'Lower side: Driver' ;
    else
        finalOverallLevels_dBA = passengerSideOverallAvg_dBA;
        finalThirdOBlevels_dBA = passengerSideThirdOctaveAvg_dBA;
        lowerSide = 'Lower side: Passenger' ;
    end

    %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    % Write FINAL results to .xlsx

    if reporting == 1 ;

        % Headers
        sheetFinalOverallHeader = strcat('Final Results Corrected for Ambient, Averaged, Lower Side: FINAL RESULT for', {' '}, year, {' '}, Make, {' '}, Model, ',', {' '}, handles.opCondSimple, ' operating condition') ;
        finalOverallHeader = {'Final Overall Results Corrected for Ambient, Averaged, Lower Side' } ;
        finalFOBHeader = {'Final FOB Results Corrected for Ambient, Averaged, Lower Side'} ;
```

```
492         lowerSideHeader = {lowerSide} ;
493
494         % Results
495         xlswrite (fileNamePath,lowerSideHeader, 'FinalSample', 'A3')
496         xlswrite (fileNamePath,finalOverallHeader, 'FinalSample', 'A4')
497         xlswrite(fileNamePath, finalOverallLevels_dBA, 'FinalSample', 'A5')
498
499         xlswrite (fileNamePath, finalFOBHeader, 'FinalSample', 'A7')
500         xlswrite (fileNamePath, oBs, 'FinalSample', 'A8')
501         xlswrite (fileNamePath, finalThirdOBlevels_dBA, 'FinalSample', 'A9')
502
503         xlswrite (fileNamePath,sheetFinalOverallHeader, 'FinalSample', 'A1')
504
505     end
506
507     %Prepare to pass data to handles to be used in the later analyses
508
509     %Frequencies up to this point are from 25-16000 Hz, need to pass 315-5000
510     %Hz for 4 band testing (NHTSA), 315-3150 Hz for 2 band (alliance)
511     NHTSAsampleData{ ink } = finalThirdOBlevels_dBA  ;
512     allianceSampleData{ ink } = finalThirdOBlevels_dBA ( 1 : 11 ) ;
513     NHTSAFront{ ink } = frontSideThirdOctaveAvg_dBA ;
514
515     micChoice{ ink } = lowerSide ;
516
517     if strcmp( lowerSide,  'Driver' )
518         fileNames.side{ ink } = driverInputFileName ;
519         fileNames.sideambient{ ink } = driverAmbientFileName ;
520         fileName.ambientSide{ ink } = resultsDriver.ambientLowerSide;
521         fileNames.side4{ ink } = resultsDriver.inputFileNames( idx );
522         fileNames.sideambient4{ ink } = resultsDriver.ambientFileNames( idx );
523         fileNames.ambientSide4{ ink } = resultsDriver.ambientLowerSide( idx );
524         fileName.frontambientSide{ ink } = { [] };
525
526     else
527         fileNames.side{ ink } = passengerInputFileName ;
528         fileNames.sideambient{ ink } = passengerAmbientFileName ;
529         fileNames.ambientSide{ ink } = resultsPassenger.ambientLowerSide;
530         fileNames.side4{ ink } = resultsPassenger.inputFileNames( idx );
531         fileNames.sideambient4{ ink } = resultsPassenger.ambientFileNames( idx );
532         fileNames.ambientSide4{ ink } = resultsPassenger.ambientLowerSide( idx );
533         fileName.frontambientSide{ ink } = { [] };
534
535     end
536
537     if ink == 2 ;
538
539         fileNames.front{ ink } = frontInputFile ;
540         fileNames.frontambient{ ink } = frontAmbientInputFile ;
541         fileNames.side { ink }= resultsFront.inputFileNames ;
```

```
542           fileNames.sideambient { ink } = resultsFront.ambientFileNames ;
543           fileNames.frontambientSide{ ink } = resultsFront.ambientLowerSide;
544           fileNames.front4{ ink } = resultsFront.inputFileNames( idx );
545           fileNames.frontambient4{ ink } = resultsFront.ambientFileNames( idx );
546           fileNames.frontambientSide4{ ink } = resultsFront.ambientLowerSide( idx );
547
548       end
549
550   end
551
552   handles.NHTSAsampleData = NHTSAsampleData;
553   handles.allianceSampleData = allianceSampleData;
554   handles.NHTSAFront = NHTSAFront;
555   handles.micChoice = micChoice;
556   handles.fileNames = fileNames;
557   handles.Make = Make;
558   handles.model = Model;
559   handles.year = year;
560
```

```
function handles = ComplianceAnalysis_GUIProcessing( handles )
% This code is used for the 4-band, 2-band, directivity, and relative volumne
% compliance testing.

set( handles.textProcessingStep, 'String', 'Analyzing Data for Compliance' );

%**************************************************************************
%**************************************************************************
% Handles conversion to variables
year = handles.year;
make = handles.Make;
model = handles.model;
char( get( handles.editNHTSANumber, 'String' ) );
vehicleNHTSAnum = char( get( handles.editNHTSANumber, 'String' ) );
vehicleVIN = char( get( handles.editVIN, 'String' ) );
userComments = {char( get( handles.editUserComments, 'String' ) ) };

NHTSAsampleData = handles.NHTSAsampleData;
allianceSampleData = handles.allianceSampleData;
micChoice = handles.micChoice;
NHTSAFront = handles.NHTSAFront;
fileNames = handles.fileNames;
numberInc = handles.numberInc;
%**************************************************************************
%**************************************************************************

%Hard code switch to turn reporting on and off.
reporting = 1;

%% LOAD REQUIREMENTS from rulingRequirementData.mat
% Final Rule, Table 6: One-Third Octave Band Minimum Requirements for Two-Band Alert
%   Vehicle Speed: Min, Band Sum
%   Reverse: 39, 48
%   < 10 km/h: 40, 44
%   >= 10 km/h, < 20 km/h: 42, 51
%   >= 20 km/h, < 30 km/h: 47, 57
%   >= 30 km/h: 52, 62
%
%Final Rule, Table 2 (Reverse), Table 1 (Stationary), Table 3 (10 km/h), Table 4 (20 km/h), Table 5 (30 km/h)
% fc: 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000
% Reverse: 42, 41, 43, 43, 44, 44, 45, 41, 42, 40, 37, 35, 33
% Stationary: 39, 39, 40, 40, 41, 41, 42, 39, 39, 37, 34, 32, 31
% 10 km/h: 45, 44, 46, 46, 47, 47, 48, 44, 45, 43, 40, 38, 36
% 20 km/h: 52, 51, 52, 53, 53, 54, 54, 51, 51, 50, 47, 45, 43
% 30 km/h: 56, 55, 57, 57, 58, 58, 59, 55, 55, 54, 51, 49, 47

load( 'rulingRequirementData.mat', 'fc' )
load( 'rulingRequirementData.mat', 'minReq2Band' )
load( 'rulingRequirementData.mat', 'minReq4Band' )

```

```
OC = [ -10 0 10 20 30 ];

%% NHTSA 4 BAND PASS TEST
% -----------------------------------------------------------------
% For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
% -----------------------------------------------------------------

%-------------------------------------------------------------------------------
%Final Rule S7.3.5:
%(d) For alerts designed to meet the four one-third octave band requirements.
% (i)Select any four one-third octave bands that are non-adjacent to each
%  other and that span a range of at least nine one-third octave bands in the
%  range of 315 Hz up to and including 5000 Hz to evaluate according to
%  subparagraph (ii), below. This step will be repeated until compliance is
%  established or it is determined that no combination meeting this selection
%  criterion can satisfy subparagraph (ii), below.
%  (ii)Compare the average corrected sound pressure level from (c) in each
%  of the four one-third octave bands selected in subparagraph (i) above to
%  the required minimum level of the corresponding one-third octave band
%  specified in paragraph S5.1.3, Table 3, to determine compliance.
%-------------------------------------------------------------------------------

fprintf( '------------------------\nComputing NHTSA Options\n------------------------\n' )

% Data Initialization
numberOfComponents = 4; % Req = 4
bandSpacing = 1; % Req = 1, minimum spacing between bands
bandRegion = 13; % Req = 13
componentSpan = 9; % Req = 9

%Create all possible combinations of bands using the above parameters
idxCombinations = getCombinations_AtLeast( numberOfComponents, bandRegion, bandSpacing, componentSpan );

%Check to make sure that a minimum span is used
if componentSpan >= numberOfComponents + ( numberOfComponents - 1 ) * bandSpacing

    % Make sure at least one combination exists
    if ~ isempty( idxCombinations )

        for ink = 1 : 5  % Looping through op types
            for chalk = 1 : size( NHTSAsampleData{ ink }, 1 )
                % Compares the signal to the requirement
                NHTSAsignalPerformance{ ink }( chalk, : ) = signalPerformance4_avg( minReq4Band{ ink },
                idxCombinations, NHTSAsampleData{ ink }( chalk, : ) );
            end
```

```
100
101             if isempty( NHTSAsignalPerformance{ ink } )
102                 NHTSAsignalPerformance{ ink } = zeros( size( idxCombinations' ) );
103             end
104         end
105     end
106 end
107
108 % Checking to see if the results from the signalPerfomance function (to
109 % compare the signal to the requirement) pass or fail
110 [ passFail, ocPass ] = passFailCheck2_Avg( NHTSAsignalPerformance );
111
112 %Report status
113 if passFail > 0
114     if passFail == 5
115         fprintf( 'The %s %s %s passed the 4-band test\n\n', year, make, model )
116     else
117         fprintf( 'The %s %s %s did not pass the 4-band test\n\n', year, make, model)
118     end
119
120     for ink = 1 : 5
121         fprintf( '%d combinations passed for %d kph operating condition\n', ocPass( ink ), OC( ink ) );
122     end
123
124     fprintf( '\n' );
125
126 else
127    fprintf( 'The %s %s %s did not pass the 4-band test\n\n', year, make, model)
128 end
129
130 %% ALLIANCE 2 BAND TEST
131
132 % -----------------------------------------------------------------
133 % For stop button
134     drawnow
135     if get(handles.pushbuttonStop, 'userdata') % stop condition
136         handles.stopButtonFlag = true ;
137         return;
138     end
139 % -----------------------------------------------------------------
140 %-------------------------------------------------------------------------------
141 %Final Rule S7.3.5:
142 % (e)For alerts designed to meet the two one-third octave band requirements.
143 %  (i)Select the two highest one-third octave bands that are non-adjacent
144 %  to - WILL "HIGHEST" WORDING BE CHANGED IN FINAL? -
145 %   each other and within the range of 315 Hz up to and including 3150Hz to
146 %   evaluate according to subparagraph (ii), below. This step will be repeated
147 %   until compliance is established or it is determined that no combination
148 %   meeting this selection criterion can satisfy paragraph (ii), below.
149 %  (ii)Compare the average corrected sound pressure level from (c) in each of
```

```
%   the two one-third octave bands selected in subparagraph (i) above to the
%   required minimum level of the corresponding one-third octave band specified
%   in paragraph S5.2 and Table 6. Also, compare the band sum of the two bands
%   to the required minimum level in Table 6.
%
%Final Rule S5.2
%One of the two bands meeting the minimum requirements in Table 6 shall be the
%band that has the highest SPL of the 315 to 800 Hz bands and the second band
%shall be the band meeting the minimum requirements in Table 6 that has the
%highest SPL of the 1000 to 3150 Hz bands. The two bands used to meet the two-band
%minimum requirements must also meet the band sum requirements as specified in Table 6.
%-------------------------------------------------------------------------------

fprintf( '--------------------------\nComputing Alliance Options\n--------------------------\n' )

% Both > 160 Hz NOTE: ANALYSIS ONLY GOES DOWN TO 315 HZ
% 1 band < 1000 Hz (315-800 Hz)
% Both < 3150 Hz (315 - 2500 Hz)

% Set up number of components, span, band region and frequency criterion
numberOfComponents = 2; % Req = 2
bandSpacing = 1; % Req = 1, minimum spacing between bands
bandRegion = 11; % Req = 11
componentSpan = 3; % Req = 3
lowFreqCriterion = 1000;

idxCombinations2 = getCombinations_band2( numberOfComponents, bandRegion, ...
    bandSpacing, fc( 1 : 11 ), lowFreqCriterion, componentSpan );

 %Check to make sure that a minimum span is used
 if componentSpan >= numberOfComponents + ( numberOfComponents - 1 ) * bandSpacing

    % Make sure at least one combination exists
    if ~ isempty( idxCombinations2 )

        for ink = 1 : 5
            for chalk = 1 : size( allianceSampleData{ ink }, 1 )
                % Compares the signal to the requirement
                [allianceSignalPerformance{ ink }( chalk, : ) bandSumTracking{ ink }( chalk, : ) ] =
                signalPerformance2_avg( minReq2Band{ ink }, idxCombinations2, allianceSampleData{ ink }( chalk, : ) );
            end

            if isempty( allianceSignalPerformance{ ink } )
                allianceSignalPerformance{ ink } = zeros( size( idxCombinations2' ) );
            end
        end
    end
 end

% Checking to see if the results from the signalPerfomance function (to
```

```
% compare the signal to the requirement) pass or fail
[ passFail2, ocPass2 ] = passFailCheck2_Avg( allianceSignalPerformance );

%Report status
if passFail2 > 0

    if passFail2 == 5
        fprintf( 'The %s %s %s passed the 2-band test\n\n', year, make, model )
    else
        fprintf( 'The %s %s %s did not pass the 2-band test\n\n', year, make, model )
    end

    for ink = 1 : 5
        fprintf( '%d combinations passed for the %d kph operating condition\n', ocPass2( ink ), OC( ink ) );
    end

    fprintf( '\n' );

else
    fprintf( 'The %s %s %s did not pass the 2-band test\n\n', year, make, model)
end


%% FINAL RESULTS
% ---------------------------------------------------------------
% For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
% ---------------------------------------------------------------

results_4band = cell( 1, 5 );
results_2band = cell( 1, 5 );

warning( 'off' ,  'MATLAB:openvar:VariableNameContainsSpaces' );
warning('off','MATLAB:xlswrite:AddSheet');

% Changing band numbers to the frequencies that the band numbers represent
for ink = 1 : 5
    for lead = 1 : size( idxCombinations, 1 )
        results_4band{ ink }( lead, 1 : 4 ) = fc( idxCombinations( lead, : ) );
    end

    % Adding 4-band test pass/fail
    for chalk = 1 : size( NHTSAsignalPerformance{ ink }, 1 )
        results_4band{ ink }( :, chalk + 4 ) = NHTSAsignalPerformance{ ink }';
    end

```

```
    % Changing band numbers to the frequencies that the band numbers represent
    for lead = 1 : size( idxCombinations2, 1 )
        results_2band{ ink }( lead, 1 : 2 ) = fc( idxCombinations2( lead, : ) );
    end

    % Adding band sum
    for frog = 1 : size( bandSumTracking{ ink }, 1 )
        results_2band{ ink }( :, chalk + 2 ) = bandSumTracking{ ink }';
    end

    % Adding 2-band test pass/fail
    for chalk = 1 : size( allianceSignalPerformance{ ink }, 1 )
        results_2band{ ink }( :, chalk + 3 ) = allianceSignalPerformance{ ink }( chalk, : )';
    end
end

% Display results for 2 and 4 band tests
if passFail == 5
    if passFail2 == 5
        fprintf( 'The %s %s %s passed the 4-band test and passed the 2-band test\n\n', year, make, model )
        passFailStatus = { 'Pass'; 'Pass' };
    else
        fprintf( 'The %s %s %s passed the 4-band test but did not pass the 2-band test\n\n', year, make, model )
        passFailStatus = { 'Pass'; 'Fail' };
    end
else
    if passFail2 == 5
        fprintf( 'The %s %s %s did not pass the 4-band test but passed the 2-band test\n\n', year, make, model )
        passFailStatus = { 'Fail'; 'Pass' };
    else
        fprintf( 'The %s %s %s did not pass the 4-band test and did not pass the 2-band test\n\n', year, make, model )
        passFailStatus = { 'Fail'; 'Fail' };
    end
end


%% DIRECTIVITY TEST
% ---------------------------------------------------------------
% For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
% ---------------------------------------------------------------

%Final Rule S5.1.1.2 For directivity, the vehicle must emit a sound
%measured at the microphone on the line CC' (FRONT MIC) having at least the A-weighted
%sound pressure level according to Table 1 in each of four non-adjacent
%bands spaning no fewer than 9 of the 13 bands from 315 to 5000Hz.
```

```
299
300  %4 band - Final Rule S5.1.1.2
301  %2-band - Final Rule S5.2.1
302  % Table 1 for Stationary (less than 10 km/h): 39, 39, 40, 40, 41, 41, 42, 39, 39, 37, 34, 32, 31
303
304  centralMicPerformance = zeros( 1, size( idxCombinations, 1 ) );
305  centralMicPerformance2 = zeros( 1, size( idxCombinations2, 1 ) );
306
307  if exist( 'NHTSAFront', 'var' )
308      centralMic  = NHTSAFront ;
309  end
310
311  if isempty( centralMic{ 2 } ) % This is hard coded to "2" becuase looking at idle condition
312      fprintf( 'This vehicle does not pass the directivity test for either the 4-band or the 2-band test\n\n');
313      directivity = { 'Fail'; 'Fail' };
314      dPass = 0;
315      dPass2 = 0;
316
317  else
318
319      if any( any( centralMic{ 2 } ) ) == 0
320          % First check if there are any nonzero values in the data (yes = 1),
321          % then check if the answer is nonzero (yes = 1), then check if the answer == 0,
322          % if no (0) then there are results, if yes (1) then there are no nonzero values
323          fprintf( 'This vehicle does not pass the directivity test for either the 4-band or the 2-band test\n\n');
324          directivity = { 'Fail'; 'Fail' };
325          dPass = 0;
326          dPass2 = 0;
327
328      else
329
330          central4bandData = centralMic{ 2 }(:, 1 : 13 );
331          central2bandData = centralMic{ 2 } (:, 1: 11 );
332
333          % 4-band test
334          for chalk = 1 : size( central4bandData, 1 )
335              centralMicPerformance( chalk, : ) = signalPerformance4_avg( minReq4Band{ 2 }, idxCombinations,
                 central4bandData( chalk , : ) );
336          end
337
338          % 2-band test
339          for chalk = 1 : size( central2bandData, 1 )
340              centralMicPerformance2( chalk, : ) = signalPerformance2_avg( minReq2Band{ 2 }, idxCombinations2,
                 central2bandData( chalk, : ) );
341          end
342
343          % Directivity test
344          [ directivityPass, dPass, dPass2 ] = directivityTest_avg( centralMicPerformance, centralMicPerformance2 );
345
346          if isempty( dPass )
```

```
347             dPass = 0;
348         end
349
350         if isempty( dPass2 )
351             dPass2 = 0;
352         end
353
354         if directivityPass == 0
355             fprintf( 'This vehicle does not pass the directivity for either the 4-band or the 2-band test\n\n');
356             directivity = { 'Fail'; 'Fail' };
357             directivityStatus = {'This vehicle does not pass the directivity for either the 4-band or the 2-band test'
                 };
358
359         elseif directivityPass == 2
360             fprintf( 'This vehicle passes the directivity test for the 4-band test but not for the 2-band test\n\n');
361             directivity = { 'Pass'; 'Fail' };
362             directivityStatus = { 'This vehicle passes the directivity test for the 4-band test but not for the
                2-band test' };
363
364         elseif directivityPass == 1
365             fprintf( 'This vehicle passes the directivity test for the 2-band test but not for the 4-band test\n\n');
366             directivity = { 'Fail'; 'Pass' };
367             directivityStatus = {'This vehicle passes the directivity test for the 2-band test but not for the 4-band
                test' };
368
369         else
370             fprintf( 'This vehicle passes the directivity test for both the 4-band test and the 2-band test\n\n');
371             directivity = { 'Pass'; 'Pass' };
372             directivityStatus = {'This vehicle passes the directivity test for both the 4-band test and the 2-band
                test' };
373
374         end
375
376     end
377
378 end
379
380 % For logging
381 directivityLabels = { 'Bands' ; 'Averaged Corrected SPL, Stationary, dBA' ; 'Minimum Requirement (Stationary, Section
    S5.1.1.2, Table 1), dBA'; 'Difference, dB' } ;
382
383 fourBandCombosLabel = {'4 Band Combinations' } ;
384 twoBandCombosLabel = {'2 Band Combinations' };
385 passFailStatusLabel = { 'Pass/Fail' } ;
386
387 bands = { '315 Hz', '400 Hz', '500 Hz', '630 Hz', '800 Hz', '1000 Hz', '1250 Hz', '1600 Hz', '2000 Hz','2500 Hz',
    '3150 Hz', '4000 Hz', '5000 Hz' } ;
388 stationarySignal = ( round( central4bandData * 10) / 10 ) ;
389 directivityThreshold = ( round( minReq4Band{ 2 } * 10) / 10 ) ;
390 directivityDifference = ( round( ( central4bandData - minReq4Band{ 2 } ) * 10 ) / 10 );
```

```
391
392  idxCombos2Bands = idxCombinationsToBands( idxCombinations ) ;
393  idxCombos2Bands2 = idxCombinationsToBands( idxCombinations2 ) ;
394
395  fourBandCombos = idxCombos2Bands ;
396  twoBandCombos = idxCombos2Bands2 ;
397
398  passFailStatus4Band = centralMicPerformance' ;
399  passFailStatus2Band = centralMicPerformance2' ;
400
401
402  %% ( VII ) RELATIVE VOLUME CHANGE
403  % ------------------------------------------------------------------
404  % For stop button
405      drawnow
406      if get(handles.pushbuttonStop, 'userdata') % stop condition
407          handles.stopButtonFlag = true ;
408          return;
409      end
410  % ------------------------------------------------------------------
411
412  % Final Rule S7.6
413  volumeChangeSample = cell( 1, 5 );
414
415  for ink = 1 : 5
416      if isempty( NHTSAsampleData{ ink } )
417          volumeChangeSample{ ink } = [ 5 5 5 5 5 5 5 5 5 5 5 5 5 ];
418      else
419          volumeChangeSample{ ink } = NHTSAsampleData{ ink };
420      end
421  end
422
423  % Relative Volume normalization and test done in volumeShift function
424  [volumeChangePass, volumeChangeIndividualStatus, handles ] = volumeShift( volumeChangeSample, minReq4Band{ 2 },
     handles );
425
426  if volumeChangePass == 1
427      fprintf( 'This vehicle passes the relative volume change test\n');
428      volumeChange = 'Pass';
429
430  else
431      fprintf( 'This vehicle did not pass the 4-band relative volume change test\n');
432      volumeChange = 'Fail';
433
434  end
435
436  % For logging
437  if handles.pass (2) == 1
438      relVolStatusStat10 = { 'This vehicle passes the relative volume test for the Stationary and 10 kmh operating
     conditions ' } ;
```

```
439   else
440       relVolStatusStat10 = { 'This vehicle does not pass the relative volume test for the Stationary and 10 kmh
          operating conditions ' } ;
441   end
442
443   if handles.pass (3) == 1
444       relVolStatus10_20 = { 'This vehicle passes the relative volume test for the 10 kmh and 20 kmh operating
          conditions ' } ;
445   else
446       relVolStatus10_20 = { 'This vehicle does not pass the relative volume test for the 10 kmh and 20 kmh operating
          conditions ' } ;
447   end
448
449   if handles.pass (4) == 1
450       relVolStatus20_30 = { 'This vehicle passes the relative volume test for the 20 kmh and 30 kmh operating
          conditions ' } ;
451   else
452       relVolStatus20_30 = { 'This vehicle does not pass the relative volume test for the 20 kmh and 30 kmh operating
          conditions ' } ;
453   end
454
455   relVolSignalLabelStat_10kmh = { 'Averaged Corrected SPL, Stationary, dBA' ; 'Averaged Corrected SPL, 10 kmh, dBA' } ;
456   relVolSignalLabel10kmh_20kmh = { 'Averaged Corrected SPL, 10 kmh, dBA' ; 'Averaged Corrected SPL, 20 kmh, dBA'  } ;
457   relVolSignalLabel20kmh_30kmh = { 'Averaged Corrected SPL, 20 kmh, dBA' ; 'Averaged Corrected SPL, 30 kmh, dBA' } ;
458
459   relVolMinReqLabel = { 'Minimum Requirement (Stationary, Section S5.1.1.2, Table 1), dBA' };
460
461   relVolNormalizedLabelStat_10kmh = { 'Normalized SPL = Stationary SPL - Requirement, dB' ; 'Normalized Signal = 10 kmh
      Level - Requirement, dB' } ;
462   relVolNormalizedLabel10kmh_20kmh = { 'Normalized SPL = 10 kmh SPL - Requirement, dB' ; 'Normalized Signal = 20 kmh
      Level - Requirement, dB' } ;
463   relVolNormalizedLabel20kmh_30kmh = { 'Normalized SPL = 20 kmh SPL - Requirement, dB' ; 'Normalized Signal = 30 kmh
      Level - Requirement, dB' } ;
464
465   relVolBandSumLabelStat_10kmh =  { 'Normalized Band Sum, Stationary, dBA' ; 'Normalized Band Sum, 10 kmh, dBA' } ;
466   relVolBandSumLabel10kmh_20kmh = { 'Normalized Band Sum, 10 kmh, dBA' ; 'Normalized Band Sum, 20 kmh, dBA' } ;
467   relVolBandSumLabel20kmh_30kmh = { 'Normalized Band Sum, 20 kmh, dBA' ; 'Normalized Band Sum, 30 kmh, dBA' } ;
468
469   relVolShiftLabelStat_10 = { 'Relative Volume Change Between Stationary and 10 kmh, dB' } ;
470   relVolShiftLabel10_20 = { 'Relative Volume Change Between 10kmh and 20 kmh, dB' } ;
471   relVolShiftLabel20_30 = { 'Relative Volume Change Between 20 kmh and 30 kmh, dB' } ;
472
473
474   relVolSignalStat = ( round( handles.meanDataSample{ 2 } * 10) / 10 ) ;
475   relVolSignal10kmh = ( round( handles.meanDataSample{ 3 } * 10) / 10 ) ;
476   relVolSignal20kmh = ( round( handles.meanDataSample{ 4 } * 10) / 10 ) ;
477   relVolSignal30kmh = ( round( handles.meanDataSample{ 5 } * 10) / 10 ) ;
478
479   relVolNormSignalStat =( round( handles.normalizedDataSample{ 2 } * 10) / 10 ) ;
480   relVolNormSignal10kmh =( round( handles.normalizedDataSample{ 3 } * 10) / 10 ) ;
```

```
relVolNormSignal20kmh =( round( handles.normalizedDataSample{ 4 } * 10) / 10 ) ;
relVolNormSignal30kmh =( round( handles.normalizedDataSample{ 5 } * 10) / 10 ) ;

relVolNormBandSumStat = ( round( handles.normalizedBandSum( 2 ) * 10) / 10 ) ;
relVolNormBandSum10kmh = ( round( handles.normalizedBandSum( 3 ) * 10) / 10 ) ;
relVolNormBandSum20kmh = ( round( handles.normalizedBandSum( 4 ) * 10) / 10 ) ;
relVolNormBandSum30kmh = ( round( handles.normalizedBandSum( 5 ) * 10) / 10 ) ;

relVolShiftStat_10kmh = ( round( handles.shift( 2 ) * 10) / 10 ) ;
relVolShift10kmh_20kmh = ( round( handles.shift( 3 ) * 10) / 10 ) ;
relVolShift20kmh_30kmh = ( round( handles.shift( 4 ) * 10) / 10 ) ;


%% Check Compliance
% ------------------------------------------------------------------
% For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
% ------------------------------------------------------------------

set( handles.textProcessingStep, 'String', 'Writing Log Files' );
set( handles.displayCurrentOC, 'Visible', 'off' );
set( handles.textRunStatusFor, 'Visible', 'off' );
drawnow

%Initialize cells
micChoiceStatus = cell( 1, 5 );
samples_numbers = cell( 1, 5 );
samples_used = cell( 1, 5 );

for ink = 1 : 5
    micChoiceStatus{ ink, 1 } = OC( ink );
    micChoiceStatus{ ink, 2 } = micChoice{ ink };
end

micChoiceSimple = strrep( micChoice, 'Lower side: ', '' ) ;

statusMicSideReverse = micChoiceSimple( 1 );
statusMicSideStationary = micChoiceSimple( 2 );
statusMicSide10kmh = micChoiceSimple( 3 );
statusMicSide20kmh = micChoiceSimple( 4 );
statusMicSide30kmh = micChoiceSimple( 5 );

%Compliance Check
status = { 'Test', '4-Band Test', '4-Band Directivity', '2-Band Test', '2-Band Directivity'; 'Status:', passFailStatus
{ 1 }, directivity{ 1 }, passFailStatus{ 2 }, directivity{ 2 }; ...
    'Operating Condition', 'Number of passing combinations','','' ,''; 'Reverse', ocPass( 1 ), '', ocPass2( 1 ), '';
```

```
        'Stationary', ocPass( 2 ), dPass, ocPass2( 2 ), dPass2 ;...
530     '10 km/h', ocPass( 3 ), '', ocPass2( 3 ), '' ; '20 km/h', ocPass( 4 ), '', ocPass2( 4 ), ''; '30 km/h', ocPass( 5
        ), '', ocPass2( 5 ), '' };
531
532 status4BandDirectivityReverse = '-' ;
533 status2BandDirectivityReverse = '-' ;
534 status4BandDirectivity10kmh = '-' ;
535 status2BandDirectivity10kmh = '-' ;
536 status4BandDirectivity20kmh = '-' ;
537 status2BandDirectivity20kmh = '-' ;
538 status4BandDirectivity30kmh = '-' ;
539 status2BandDirectivity30kmh = '-' ;
540 relVolNormBandSumRev = '-' ;
541
542 relVolNormBandSumStatRound = num2str( ( round( relVolNormBandSumStat * 10) / 10 ) );
543 relVolNormBandSum10kmhRound = num2str( ( round( relVolNormBandSum10kmh * 10) / 10 ) ) ;
544 relVolNormBandSum20kmhRound = num2str( ( round( relVolNormBandSum20kmh * 10) / 10 ) );
545 relVolNormBandSum30kmhRound = num2str( ( round( relVolNormBandSum30kmh * 10) / 10 ) );
546
547 ocPassStr = arrayfun(@num2str,ocPass, 'un', 0 ) ;
548 ocPass2Str = arrayfun(@num2str,ocPass2, 'un', 0 ) ;
549 dPassStr = num2str( dPass ) ;
550 dPass2Str = num2str( dPass2 ) ;
551
552 handles.datStatus = [ statusMicSideReverse, statusMicSideStationary, statusMicSide10kmh, statusMicSide20kmh,
    statusMicSide30kmh ; ...
553     ocPassStr( 1 ), ocPassStr( 2 ), ocPassStr( 3 ), ocPassStr( 4 ), ocPassStr( 5 ); ...
554     { status4BandDirectivityReverse }, { dPassStr  }, { status4BandDirectivity10kmh }, { status4BandDirectivity20kmh
        }, { status4BandDirectivity30kmh }; ...
555     ocPass2Str( 1 ), ocPass2Str( 2 ), ocPass2Str( 3 ), ocPass2Str( 4 ), ocPass2Str( 5 ); ...
556     { status2BandDirectivityReverse }, { dPass2Str }, { status2BandDirectivity10kmh }, { status2BandDirectivity20kmh
        }, { status2BandDirectivity30kmh }; ...
557     { relVolNormBandSumRev }, { relVolNormBandSumStatRound }, { relVolNormBandSum10kmhRound }, {
        relVolNormBandSum20kmhRound }, { relVolNormBandSum30kmhRound }] ;
558
559 % Assumes all tests fail first
560 passTestOC = [ 0 0 0 0 0 ];
561 passTestDir = 0;
562 passTestVol = 0;
563
564 % Passes the operating condition (OC) test if any OC passes EITHER the 2 OR 4 band test
565 cnt = 1;
566 for ink = 1 : 5
567     if ocPass( ink ) || ocPass2( ink )
568         passTestOC( ink ) = 1;
569     else
570         failsOC( cnt ) = ink;
571         cnt = cnt + 1;
572     end
573 end
```

```
574
575   % If any of the OCs fail BOTH the 2 and 4 band test, the vehicle fails
576   if any( passTestOC == 0 )
577       passTestallOCs = 0 ;
578   else
579       passTestallOCs = 1 ;
580   end
581
582   % Passes the directivity test if EITHER 2 OR 4 band directivity passes
583   if dPass || dPass2
584       passTestDir = 1;
585   else
586           failsDir( cnt ) = ink + 1;
587           cnt = cnt + 1;
588   end
589
590   % Must pass relative volume (volume shift) test for 0kmh to 10kmh, 10kmh to 20kmh, AND 20kmh to 30kmh
591   if sum( volumeChangeIndividualStatus ) == 3
592       passTestVol = 1;
593   else
594       failsVol( cnt ) = ink + 2;
595       cnt = cnt + 1;
596   end
597
598   % If any of the tests (OC, directivity, relative volume) fail, the vehicle
599   % is noncompliant
600   if passTestallOCs && passTestDir && passTestVol
601       handles.complianceOverall = 'Pass' ;
602   else
603       handles.complianceOverall = 'Fail' ;
604   end
605
606   %% Write excel files
607   % -----------------------------------------------------------------
608   % For stop button
609       drawnow
610       if get(handles.pushbuttonStop, 'userdata') % stop condition
611           handles.stopButtonFlag = true ;
612           return;
613       end
614   % -----------------------------------------------------------------
615
616   opCondOptions = ({ 'Reverse', 'Stationary', '10kmh', '20kmh', '30kmh' }) ;
617
618   if reporting == 1 ;
619
620       for ink = 1 : 5
621           ocName = opCondOptions{ ink } ;
622           yearStr = num2str( year ) ;
623
```

```
624         % Vehicle and Test Site Info
625         fileName = sprintf('%s%s%s_%s_%d.xlsx',year, make, model, ocName, numberInc );
626         fileNamePath = fullfile( handles.outputDirectory, fileName) ;
627
628         vehicle_info = { 'Vehicle Information', ''; 'Year', yearStr; 'Make', make; 'Model', model; 'NHTSA #',
            vehicleNHTSAnum; 'VIN', vehicleVIN };
629         xlswrite( fileNamePath, vehicle_info, 'Sheet1', 'A5' );
630
631         %Operating condition and lower side
632         oc_info = {'Operating Condition', ocName };
633         xlswrite( fileNamePath, oc_info, 'Sheet1', 'A12' );
634         micOC = micChoiceSimple{ ink } ;
635         mic_info = {'Lower Side', micOC } ;
636         xlswrite( fileNamePath, mic_info, 'Sheet1', 'A13' );
637
638         samples_tested_headers = { 'Samples Input for Testing ( Lower Side )' };
639         xlswrite( fileNamePath, samples_tested_headers, 'Sheet1', 'A13' );
640         xlswrite( fileNamePath, fileNames.side { ink }, 'Sheet1', 'B13' );
641
642         samples_testedAmb_headers = { 'Samples Input for Testing ( Lower Side Ambient )' };
643         xlswrite( fileNamePath, samples_testedAmb_headers, 'Sheet1', 'A15' );
644         xlswrite( fileNamePath, fileNames.sideambient { ink }, 'Sheet1', 'B15' );
645         xlswrite( fileNamePath, fileNames.ambientSide { ink }, 'Sheet1', 'B16' );
646
647         samples_testedFront_headers = { 'Samples Input for Testing ( Front )' };
648         xlswrite( fileNamePath, samples_testedFront_headers, 'Sheet1', 'A18' );
649         xlswrite( fileNamePath, { 'NA' }, 'Sheet1', 'B18' );
650
651         samples_testedFrontAmb_headers = { 'Samples Input for Testing ( Front Ambient )' };
652         xlswrite( fileNamePath, samples_testedFrontAmb_headers, 'Sheet1', 'A20' );
653         xlswrite( fileNamePath, { 'NA' }, 'Sheet1', 'B20' );
654
655         if ink == 2
656             xlswrite( fileNamePath, fileNames.front { ink }, 'Sheet1', 'B18' );
657             xlswrite( fileNamePath, fileNames.frontambient { ink }, 'Sheet1', 'B20' );
658             xlswrite( fileNamePath, fileNames.frontambientSide { ink }, 'Sheet1', 'B21' );
659         end
660
661         samples_passing_2dB_headers = { 'Samples Used in 4 Band and 2 Band Tests ( Lower Side )' };
662         xlswrite( fileNamePath, samples_passing_2dB_headers, 'Sheet1', 'A23' );
663         xlswrite( fileNamePath, fileNames.side4 { ink }, 'Sheet1', 'B23' );
664
665         samples_passing_2dB_headers_amb = { 'Samples Used in 4 Band and 2 Band Tests ( Lower Side Ambient )' };
666         xlswrite( fileNamePath, samples_passing_2dB_headers_amb, 'Sheet1', 'A25' );
667         xlswrite( fileNamePath, fileNames.sideambient4 { ink }, 'Sheet1', 'B25' );
668         xlswrite( fileNamePath, fileNames.ambientSide4 { ink }, 'Sheet1', 'B26' );
669
670         samples_passing_2dB_headers_front = { 'Samples Used in 4 Band and 2 Band Tests ( Front )' };
671         xlswrite( fileNamePath, samples_passing_2dB_headers_front, 'Sheet1', 'A28' );
672         xlswrite( fileNamePath, { 'NA' }, 'Sheet1', 'B28' );
```

```
673
674         samples_passing_2dB_headers_frontAmb = { 'Samples Used in 4 Band and 2 Band Tests ( Front Ambient )' };
675         xlswrite( fileNamePath, samples_passing_2dB_headers_frontAmb, 'Sheet1', 'A30' );
676         xlswrite( fileNamePath, { 'NA' }, 'Sheet1', 'B30' );
677
678         if ink == 2
679             xlswrite( fileNamePath, fileNames.front4 { ink }, 'Sheet1', 'B28' );
680             xlswrite( fileNamePath, fileNames.frontambient4 { ink }, 'Sheet1', 'B30' );
681             xlswrite( fileNamePath, fileNames.frontambientSide4 { ink }, 'Sheet1', 'B31' );
682         end
683
684         idxvalidTrue = cell( 1, 5 );
685
686         headers_4band = { '4-Band Combination', '', '', '', 'Pass=1/Fail=0';} ;
687         headers_2band = { '2-Band Combination', '', 'Band Sum for Passing Combinations', 'Pass=1/Fail=0'; };
688         header_bandsum = { 'Band Sum for Passing Combinations' }; % For directivity
689
690         if ink == 1 ;
691             % Reverse Operating Condition
692             reverseSheetHeader = {'Reverse Condition Samples - Band Combinations' } ;
693
694             xlswrite( fileNamePath, headers_4band, 'Reverse', 'A5' );
695             xlswrite( fileNamePath, headers_2band, 'Reverse', 'H5' );
696
697             xlswrite( fileNamePath, results_4band{ 1 }, 'Reverse', 'A6' );
698             xlswrite( fileNamePath, results_2band{ 1 }, 'Reverse', 'H6' );
699
700             xlswrite( fileNamePath, reverseSheetHeader, 'Reverse', 'A1' );
701         end
702
703         if ink == 2 ;
704
705             % Stationary Operating Condition
706             StationarySheetHeader = {'Stationary Condition Samples - Band Combinations' } ;
707             DirectivitySheetHeader = {'Directivity Results' } ;
708
709             xlswrite( fileNamePath, headers_4band, 'Stationary', 'A5' );
710             xlswrite( fileNamePath, headers_2band, 'Stationary', 'H5' );
711
712             xlswrite( fileNamePath, results_4band{ 2 }, 'Stationary', 'A6' );
713             xlswrite( fileNamePath, results_2band{ 2 }, 'Stationary', 'H6' );
714
715             xlswrite( fileNamePath, StationarySheetHeader, 'Stationary', 'A1' );
716
717
718             xlswrite( fileNamePath, directivityStatus, 'Directivity', 'A3' );
719
720             xlswrite( fileNamePath, directivityLabels, 'Directivity', 'A5' );
721             xlswrite( fileNamePath, bands, 'Directivity', 'B5' );
722             xlswrite( fileNamePath, stationarySignal, 'Directivity', 'B6' );
```

```
723             xlswrite( fileNamePath, directivityThreshold, 'Directivity', 'B7' );
724             xlswrite( fileNamePath, directivityDifference, 'Directivity', 'B8' );
725 
726             xlswrite( fileNamePath, fourBandCombosLabel, 'Directivity', 'A11' );
727             xlswrite( fileNamePath, twoBandCombosLabel, 'Directivity', 'H11' );
728             xlswrite( fileNamePath, passFailStatusLabel, 'Directivity', 'E11' );
729             xlswrite( fileNamePath, header_bandsum, 'Directivity', 'J11' ) ;
730             xlswrite( fileNamePath, passFailStatusLabel, 'Directivity', 'K11' );
731 
732             xlswrite( fileNamePath, fourBandCombos, 'Directivity', 'A12' );
733             xlswrite( fileNamePath, twoBandCombos, 'Directivity', 'H12' );
734             xlswrite( fileNamePath, passFailStatus4Band, 'Directivity', 'E12' );
735             xlswrite( fileNamePath, bandSumTracking{ 2 }', 'Directivity', 'J12' ) ;
736             xlswrite( fileNamePath, passFailStatus2Band, 'Directivity', 'K12' );
737 
738 
739             xlswrite( fileNamePath, DirectivitySheetHeader, 'Directivity', 'A1' );
740 
741         end
742 
743         if ink == 3 ;
744 
745             % 10 kph Operating Condition
746             tenkmhSheetHeader = {'10 kmh Condition Samples - Band Combinations' } ;
747             relVolSheetHeader = {' Relative Volume Results' } ;
748 
749             xlswrite( fileNamePath, fileNames.side ( 3 ), '10_kmh', 'B3' );
750 
751             xlswrite( fileNamePath, headers_4band, '10_kmh', 'A5' );
752             xlswrite( fileNamePath, headers_2band, '10_kmh', 'H5' );
753 
754             xlswrite( fileNamePath, results_4band{ 3  }, '10_kmh', 'A6' );
755             xlswrite( fileNamePath, results_2band{ 3 }, '10_kmh', 'H6' );
756 
757             xlswrite( fileNamePath, tenkmhSheetHeader, '10_kmh', 'A1' );
758 
759             xlswrite( fileNamePath, bands, 'RelativeVolume', 'B5' );
760             xlswrite( fileNamePath, relVolStatusStat10, 'RelativeVolume', 'A3' );
761             xlswrite( fileNamePath, relVolSignalLabelStat_10kmh, 'RelativeVolume', 'A6' );
762             xlswrite( fileNamePath, relVolSignalStat, 'RelativeVolume', 'B6' );
763             xlswrite( fileNamePath, relVolSignal10kmh, 'RelativeVolume', 'B7' );
764 
765             xlswrite( fileNamePath, relVolMinReqLabel, 'RelativeVolume', 'A9' );
766             xlswrite( fileNamePath, handles.RelVolMinReq, 'RelativeVolume', 'B9' );
767 
768             xlswrite( fileNamePath, relVolNormalizedLabelStat_10kmh, 'RelativeVolume', 'A11' );
769             xlswrite( fileNamePath, relVolNormSignalStat, 'RelativeVolume', 'B11' );
770             xlswrite( fileNamePath, relVolNormSignal10kmh, 'RelativeVolume', 'B12' );
771 
772             xlswrite( fileNamePath, relVolBandSumLabelStat_10kmh, 'RelativeVolume', 'A14' );
```

```
773             xlswrite( fileNamePath, relVolNormBandSumStat, 'RelativeVolume', 'B14' );
774             xlswrite( fileNamePath, relVolNormBandSum10kmh, 'RelativeVolume', 'B15' );
775
776             xlswrite( fileNamePath, relVolShiftLabelStat_10, 'RelativeVolume', 'A17' );
777             xlswrite( fileNamePath, relVolShiftStat_10kmh, 'RelativeVolume', 'B17' );
778
779             xlswrite( fileNamePath, relVolSheetHeader, 'RelativeVolume', 'A1' );
780
781
782         end
783
784         if ink == 4 ;
785
786             % 20 kph Operating Condition
787             twentykmhSheetHeader = {'20 kmh Condition Samples - Band Combinations' } ;
788
789             xlswrite( fileNamePath, fileNames.side ( 4 ), '20_kmh', 'B3' );
790
791             xlswrite( fileNamePath, headers_4band, '20_kmh', 'A5' );
792             xlswrite( fileNamePath, headers_2band, '20_kmh', 'H5' );
793
794             xlswrite( fileNamePath, results_4band{ 4  }, '20_kmh', 'A6' );
795             xlswrite( fileNamePath, results_2band{ 4 }, '20_kmh', 'H6' );
796
797             xlswrite( fileNamePath, twentykmhSheetHeader, '20_kmh', 'A1' );
798
799             xlswrite( fileNamePath, bands, 'RelativeVolume', 'B5' );
800             xlswrite( fileNamePath, relVolStatus10_20, 'RelativeVolume', 'A3' );
801             xlswrite( fileNamePath, relVolSignalLabel10kmh_20kmh, 'RelativeVolume', 'A6' );
802             xlswrite( fileNamePath, relVolSignal10kmh, 'RelativeVolume', 'B6' );
803             xlswrite( fileNamePath, relVolSignal20kmh, 'RelativeVolume', 'B7' );
804
805             xlswrite( fileNamePath, relVolMinReqLabel, 'RelativeVolume', 'A9' );
806             xlswrite( fileNamePath, handles.RelVolMinReq, 'RelativeVolume', 'B9' );
807
808             xlswrite( fileNamePath, relVolNormalizedLabel10kmh_20kmh, 'RelativeVolume', 'A11' );
809             xlswrite( fileNamePath, relVolNormSignal10kmh, 'RelativeVolume', 'B11' );
810             xlswrite( fileNamePath, relVolNormSignal20kmh, 'RelativeVolume', 'B12' );
811
812             xlswrite( fileNamePath, relVolBandSumLabel10kmh_20kmh, 'RelativeVolume', 'A14' );
813             xlswrite( fileNamePath, relVolNormBandSum10kmh, 'RelativeVolume', 'B14' );
814             xlswrite( fileNamePath, relVolNormBandSum20kmh, 'RelativeVolume', 'B15' );
815
816             xlswrite( fileNamePath, relVolShiftLabel10_20, 'RelativeVolume', 'A17' );
817             xlswrite( fileNamePath, relVolShift10kmh_20kmh, 'RelativeVolume', 'B17' );
818
819             xlswrite( fileNamePath, relVolSheetHeader, 'RelativeVolume', 'A1' );
820
821         end
822
```

```
823         if ink == 5 ;
824
825             % 30 kph Operating Condition
826             thirtykmhSheetHeader = strcat('30 kmh Condition Samples - Band Combinations for', {' '}, year, {' '}, make
                , {' '}, model, ',', {' '}, handles.opCondSimple, ' operating condition') ;
827             xlswrite( fileNamePath, fileNames.side ( 5 ), '30_kmh', 'B3' );
828
829             xlswrite( fileNamePath, headers_4band, '30_kmh', 'A5' );
830             xlswrite( fileNamePath, headers_2band, '30_kmh', 'H5' );
831
832             xlswrite( fileNamePath, results_4band{ 5 }, '30_kmh', 'A6' );
833             xlswrite( fileNamePath, results_2band{ 5 }, '30_kmh', 'H6' );
834
835             xlswrite( fileNamePath, thirtykmhSheetHeader, '30_kmh', 'A1' );
836
837             xlswrite( fileNamePath, bands, 'RelativeVolume', 'B5' );
838             xlswrite( fileNamePath, relVolStatus20_30, 'RelativeVolume', 'A3' );
839             xlswrite( fileNamePath, relVolSignalLabel20kmh_30kmh, 'RelativeVolume', 'A6' );
840             xlswrite( fileNamePath, relVolSignal20kmh, 'RelativeVolume', 'B6' );
841             xlswrite( fileNamePath, relVolSignal30kmh, 'RelativeVolume', 'B7' );
842
843             xlswrite( fileNamePath, relVolMinReqLabel, 'RelativeVolume', 'A9' );
844             xlswrite( fileNamePath, handles.RelVolMinReq, 'RelativeVolume', 'B9' );
845
846             xlswrite( fileNamePath, relVolNormalizedLabel20kmh_30kmh, 'RelativeVolume', 'A11' );
847             xlswrite( fileNamePath, relVolNormSignal20kmh, 'RelativeVolume', 'B11' );
848             xlswrite( fileNamePath, relVolNormSignal30kmh, 'RelativeVolume', 'B12' );
849
850             xlswrite( fileNamePath, relVolBandSumLabel20kmh_30kmh, 'RelativeVolume', 'A14' );
851             xlswrite( fileNamePath, relVolNormBandSum20kmh, 'RelativeVolume', 'B14' );
852             xlswrite( fileNamePath, relVolNormBandSum30kmh, 'RelativeVolume', 'B15' );
853
854             xlswrite( fileNamePath, relVolShiftLabel20_30, 'RelativeVolume', 'A17' );
855             xlswrite( fileNamePath, relVolShift20kmh_30kmh, 'RelativeVolume', 'B17' );
856
857             xlswrite( fileNamePath, relVolSheetHeader, 'RelativeVolume', 'A1' );
858
859         end
860
861         fileName = sprintf('%s%s%s_%s_%d.xlsx',year, make, model, ocName, numberInc );
862         fileNamePath = fullfile( handles.outputDirectory, fileName) ;
863
864         vehicle_info = { 'Vehicle Information', ''; 'Year', yearStr; 'Make', make; 'Model', model; 'NHTSA #',
            vehicleNHTSAnum; 'VIN', vehicleVIN };
865         xlswrite( fileNamePath, vehicle_info, 'Sheet1', 'A5' );
866
867         introHeader1 = {'Minimum Sound Requirements for Hybrid and Electric Vehicles' } ;
868         introHeader21 = 'Compliance Tool Version ' ;
869         introHeader22 = handles.versionNumber ;
870         introHeader2 = { sprintf('%s%s', introHeader21, introHeader22)} ;
```

```
871          introHeader3 = {'Log File'} ;
872
873          xlswrite (fileNamePath,introHeader1, 'Sheet1', 'A1')
874          xlswrite (fileNamePath,introHeader2, 'Sheet1', 'A2')
875          xlswrite (fileNamePath,introHeader3, 'Sheet1', 'A3')
876
877      end
878
879  %% Prep variables for summary log
880
881  vehicleInfo = [{ year } ; { make } ; { model } ; { vehicleNHTSAnum } ; { vehicleVIN } ] ;
882  compliaceOverallResult = { handles.complianceOverall } ;
883
884  if passTestDir == 1
885      passTestDirPF = 'Pass' ;
886  else
887      passTestDirPF = 'Fail' ;
888  end
889
890  for ink = 1 : 5
891      if ocPass( ink ) > 0
892          ocPassStatus{ ink } = 'Pass';
893      else
894          ocPassStatus{ ink } = 'Fail';
895      end
896  end
897
898  for ink = 1 : 5
899      if ocPass2( ink ) > 0
900          ocPassStatus2{ ink } = 'Pass';
901      else
902          ocPassStatus2{ ink } = 'Fail';
903      end
904  end
905
906  for ink = 1 : 5
907      if ocPass( ink ) || ocPass2( ink )
908          ocPassOverall{ ink } = 'Pass';
909      else
910          ocPassOverall{ ink } = 'Fail';
911      end
912  end
913
914  fourFileNamesReverse = fileNames.side4 { 1 } ;
915  fourFileNamesStationary = fileNames.side4 { 2 } ;
916  fourFileNames10kmh = fileNames.side4 { 3 } ;
917  fourFileNames20kmh = fileNames.side4 { 4 } ;
918  fourFileNames30kmh = fileNames.side4 {5 } ;
919
920  volChangeStatusConvert2str = strread(num2str( volumeChangeIndividualStatus ), '%s' ) ;
```

```
volChangeStatusReplaceFail = strrep(volChangeStatusConvert2str, '0', 'Fail') ;
volChangeStatusPassFail = strrep(volChangeStatusReplaceFail, '1', 'Pass') ;
statusVolumeChangeStat_10kmh =  volChangeStatusPassFail(2)  ;
statusVolumeChange10kmh_20kmh = volChangeStatusPassFail(3) ;
statusVolumeChange20kmh_30kmh = volChangeStatusPassFail(4)  ;

datStatusSummaryLog = [  ocPassStr( 2 ), ocPassStatus( 2 ),  ocPass2Str( 2 ), ocPassStatus2( 2 ), ocPassOverall( 2 ), ' ', statusMicSideStationary, fourFileNamesStationary; ...
    { dPassStr  }, directivity{ 1 }, { dPass2Str }, directivity{ 2 }, passTestDirPF, ' ', 'Front', fileNames.front4{ 2 };...
    ocPassStr( 1 ), ocPassStatus( 1 ),  ocPass2Str( 1 ), ocPassStatus2( 1 ), ocPassOverall( 1 ), ' ', statusMicSideReverse, fourFileNamesReverse; ...
    ocPassStr( 3 ), ocPassStatus( 3 ),  ocPass2Str( 3 ), ocPassStatus2( 3 ), ocPassOverall( 3 ), statusVolumeChangeStat_10kmh, statusMicSide10kmh, fourFileNames10kmh; ...
    ocPassStr( 4 ), ocPassStatus( 4 ),  ocPass2Str( 4 ), ocPassStatus2( 4 ), ocPassOverall( 4 ), statusVolumeChange10kmh_20kmh, statusMicSide20kmh, fourFileNames20kmh; ...
    ocPassStr( 5 ), ocPassStatus( 5 ),  ocPass2Str( 5 ), ocPassStatus2( 5 ), ocPassOverall( 5 ), statusVolumeChange20kmh_30kmh, statusMicSide30kmh, fourFileNames30kmh ] ;

%% Write Summary Log
fileNameSummary = sprintf('%s%s%s_Summary_%d.xlsx',year, make, model, numberInc);
fileNameSummaryPath = fullfile( handles.outputDirectory, fileNameSummary) ;

copyfile('BaseData.bin', fileNameSummaryPath)

summaryVersionNumber11 = 'Quiet Vehicle Compliance Tool Version: ' ;
summaryVersionNumber12 = handles.versionNumber ;
summaryVersionNumber1 = {sprintf('%s%s', summaryVersionNumber11, summaryVersionNumber12)} ;

xlswrite(fileNameSummaryPath, summaryVersionNumber1, 'Sheet1', 'A19')
xlswrite(fileNameSummaryPath, datStatusSummaryLog, 'Sheet1', 'F7')
xlswrite(fileNameSummaryPath, compliaceOverallResult, 'Sheet1', 'J17')

xlswrite(fileNameSummaryPath, vehicleInfo, 'Sheet1', 'B7')
xlswrite(fileNameSummaryPath, userComments, 'Sheet1', 'A14')
xlswrite(fileNameSummaryPath, ' ', 'Sheet1', 'A3')

end
```

```
function pA = Compute_A_Weighting_44p1_kHz( pZ, B, A )

% This function computes A-weighted sound pressure, pa
% Filter coefficients designed for fs = 44.1 kHz
pA = filter( B, A, pZ );


```

```
function [ spl, t ] = Compute_Fast_Time_Weighting_44p1_kHz( pA )
% This function computes sound pressure level with fast time weighting
% Time weighting for fs = 44.1 kHz

fs = 44100;

Tau = 125 / 1000; % 125 mili-seconds

N = 10; % Number of Taus that are included in averaging
Ntau = fs * Tau; % Number of sample points in one Tau, used as scaling factor
time4Window = 0 : 1 / fs : N * Tau; % Time for the window
weight = exp( -time4Window / Tau ); % Time window

pref2 = ( 20e-6 )^2;

% pA = A-weighted pressure; it is the input to this function
% B = weight, by definition time weighting applied to pressure squared
% fftfilt = To convolve in the time domain, multiply in the frequency
% domain, i.e. use fftflit. The greater the number of samples in "weight",
% the greater the result will be. Therefore, scale by Ntau.
pAFastAvg = fftfilt( weight, pA.^2 ) / Ntau;

% Data should not be negative (squared real)
% Replace 0 with smallest possible increment
pAFastAvg( pAFastAvg <= 0 ) = eps(0);

spl = 10 * log10( pAFastAvg ) - 10 * log10 ( pref2 );

t = ( 0 : length( spl ) - 1 ) / fs;
```

```
function oneThirdOBLevels = Compute_One_Third_Octave_Bands_44p1_kHz_Hd( Hd, pA )
% oneThirdOctaveBandLevel = each row is a 1/3 OB filtered version of the
% original time signal

oneThirdOBLevels( 1, : ) = filter( Hd( 1 ), pA );
oneThirdOBLevels = zeros( length( Hd ), length( oneThirdOBLevels ) );

% Filter
% for ink = 1 : length( Hd )
%     oneThirdOBLevels( ink, : ) = filter( Hd( ink ), pA );
% end

% Filter
for ink = 1 : length( Hd )
    [ b, a ] = tf( Hd( ink ) ) ;
    oneThirdOBLevels( ink, : ) = filter( b, a, pA );
end

```

```
function [ handles ] = Compute_SPL_by_Vehicle_Side_forGUI( handles )
% This function computes sound pressure level for each microphone (driver,
% passenger, front, and trigger)

cnt = 1 ;

numSamplesLengthForLoop = length(unique( handles.data( handles.data.opCond == handles.opCondForLoop, { 'sampleID' } )
) );

% Pull all samples for current operating condition only
driverDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide == 'driver' &
handles.data.signalType == 'signal', :) );
preAmbDriverDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide == 'driver'
 & handles.data.signalType == 'preAmbient', :) );
postAmbDriverDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide ==
'driver' & handles.data.signalType == 'postAmbient', :) );

passengerDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide == 'passenger'
 & handles.data.signalType == 'signal', :) );
preAmbPassengerDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide ==
'passenger' & handles.data.signalType == 'preAmbient', :) );
postAmbPassengerDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide ==
'passenger' & handles.data.signalType == 'postAmbient', :) );

frontDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide == 'front' &
handles.data.signalType == 'signal', :) );
preAmbFrontDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide == 'front' &
 handles.data.signalType == 'preAmbient', :) );
postAmbFrontDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide == 'front'
& handles.data.signalType == 'postAmbient', :) );

triggerDataList = ( handles.data( handles.data.opCond == handles.opCondForLoop & handles.data.micSide == 'all' &
handles.data.signalType == 'trigger', :) );

%% Display current operating condition and total operations for current operating condition
set( handles.textProcessingStep, 'String', 'Processing sample' );
set( handles.displayRunStatusCurrentFile, 'Visible', 'on' );
set( handles.textRunStatusOf, 'Visible', 'on');
set( handles.displayRunStatusTotalFiles, 'Visible', 'on' );
set( handles.textRunStatusFor, 'Visible', 'on');
set( handles.displayCurrentOC, 'Visible', 'on');

totalSamplesforOC = num2str( length( driverDataList ) );
set( handles.displayRunStatusTotalFiles, 'String', totalSamplesforOC );
set( handles.displayCurrentOC, 'String', handles.opCondForLoop );
drawnow

%% Prep GUI inputs and run through SLM for current OC for all samples

for cnt = 1 : numSamplesLengthForLoop
```

```
40
41      % ------------------------------------------------------------------
42      % For stop button
43      drawnow
44      if get(handles.pushbuttonStop, 'userdata') % stop condition
45          handles.stopButtonFlag = 'true' ;
46          return;
47      end
48      % ------------------------------------------------------------------
49
50      set( handles.displayRunStatusCurrentFile, 'String', num2str( cnt ) );
51      drawnow
52
53      % Prep GUI inputs for input into SLM
54
55      % Driver signal, pre, and post ambient
56      driver.inputFileName = char( driverDataList.fileName( cnt ) );
57      driver.inputPathName = char( driverDataList.pathName( cnt ) );
58      [driverData, fs] = wavread( [  driver.inputPathName driver.inputFileName ] ) ;
59      driverData = driverData( : , driverDataList.channel( cnt ) );
60      if fs ~= 44100 % Resample if needed
61          driverData = resample( driverData, 44100, fs ); %fs is not used after this, do not need to reassign to 44100
62      end
63
64      preAmbDriver.inputFileName = char( preAmbDriverDataList.fileName( cnt ) );
65      preAmbDriver.inputPathName = char( preAmbDriverDataList.pathName( cnt ) );
66      [preAmbDriverData, fs] = wavread( [  preAmbDriver.inputPathName preAmbDriver.inputFileName ] ) ;
67      preAmbDriverData = preAmbDriverData( : , preAmbDriverDataList.channel( cnt ) );
68      if fs ~= 44100 % Resample if needed
69          preAmbDriverData = resample( preAmbDriverData, 44100, fs );
70      end
71
72      postAmbDriver.inputFileName = char( postAmbDriverDataList.fileName( cnt ) );
73      postAmbDriver.inputPathName = char( postAmbDriverDataList.pathName( cnt ) );
74      [postAmbDriverData, fs] = wavread( [  postAmbDriver.inputPathName postAmbDriver.inputFileName ] ) ;
75      postAmbDriverData = postAmbDriverData( : , postAmbDriverDataList.channel( cnt ) );
76      if fs ~= 44100 % Resample if needed
77          postAmbDriverData = resample( postAmbDriverData, 44100, fs );
78      end
79
80      % Passenger signal, pre, and post ambient
81      passenger.inputFileName = char( passengerDataList.fileName( cnt ) );
82      passenger.inputPathName = char( passengerDataList.pathName( cnt ) );
83      [passengerData, fs] = wavread( [  passenger.inputPathName passenger.inputFileName ] ) ;
84      passengerData = passengerData( : , passengerDataList.channel( cnt ) );
85      if fs ~= 44100 % Resample if needed
86          passengerData = resample( passengerData, 44100, fs );
87      end
88
89      preAmbPassenger.inputFileName = char( preAmbPassengerDataList.fileName( cnt ) );
```

```
90        preAmbPassenger.inputPathName = char( preAmbPassengerDataList.pathName( cnt ) );
91        [preAmbPassengerData, fs] = wavread( [  preAmbPassenger.inputPathName preAmbPassenger.inputFileName ] ) ;
92        preAmbPassengerData = preAmbPassengerData( : , preAmbPassengerDataList.channel( cnt ) );
93        if fs ~= 44100 % Resample if needed
94            preAmbPassengerData = resample( preAmbPassengerData, 44100, fs );
95        end
96
97        postAmbPassenger.inputFileName = char( postAmbPassengerDataList.fileName( cnt ) );
98        postAmbPassenger.inputPathName = char( postAmbPassengerDataList.pathName( cnt ) );
99        [postAmbPassengerData, fs] = wavread( [  postAmbPassenger.inputPathName postAmbPassenger.inputFileName ] ) ;
100       postAmbPassengerData = postAmbPassengerData( : , postAmbPassengerDataList.channel( cnt ) );
101       if fs ~= 44100 % Resample if needed
102           postAmbPassengerData = resample( postAmbPassengerData, 44100, fs );
103       end
104
105       if strcmp(handles.opCondForLoop, 'Stationary')
106           front.inputFileName = char( frontDataList.fileName( cnt ) );
107           front.inputPathName = char( frontDataList.pathName( cnt ) );
108           [frontData, fs] = wavread( [  front.inputPathName front.inputFileName ] ) ;
109           frontData = frontData( : , frontDataList.channel( cnt ) );
110           if fs ~= 44100 % Resample if needed
111               frontData = resample( frontData, 44100, fs );
112           end
113
114           preAmbFront.inputFileName = char( preAmbFrontDataList.fileName( cnt ) );
115           preAmbFront.inputPathName = char( preAmbFrontDataList.pathName( cnt ) );
116           [preAmbFrontData, fs] = wavread( [  preAmbFront.inputPathName preAmbFront.inputFileName ] ) ;
117           preAmbFrontData = preAmbFrontData( : , preAmbFrontDataList.channel( cnt ) );
118           if fs ~= 44100 % Resample if needed
119               preAmbFrontData = resample( preAmbFrontData, 44100, fs );
120           end
121
122           postAmbFront.inputFileName = char( postAmbFrontDataList.fileName( cnt ) );
123           postAmbFront.inputPathName = char( postAmbFrontDataList.pathName( cnt ) );
124           [postAmbFrontData, fs] = wavread( [  postAmbFront.inputPathName postAmbFront.inputFileName ] ) ;
125           postAmbFrontData = postAmbFrontData( : , postAmbFrontDataList.channel( cnt ) );
126           if fs ~= 44100 % Resample if needed
127               postAmbFrontData = resample( postAmbFrontData, 44100, fs );
128           end
129
130       end
131
132       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
133
134       % Compute SPL for Single Channel
135
136       % ---------------------------------------------------------------
137       % For stop button
138       drawnow
139       if get(handles.pushbuttonStop, 'userdata') % stop condition
```

```
140         handles.stopButtonFlag = 'true' ;
141         return;
142     end
143     % -----------------------------------------------------------------
144
145     % For passbys, trim driver/passenger singal data at trigger time
146     if strcmp(handles.opCondForLoop,'10 km/h' ) || strcmp (handles.opCondForLoop,'20 km/h' )||  strcmp (handles.
        opCondForLoop,'30 km/h' );
147
148         % Resample trigger data
149         trigger.inputFileName = char( triggerDataList.fileName( cnt ) );
150         trigger.inputPathName = char( triggerDataList.pathName( cnt ) );
151         [triggerData, fs] = wavread( [  trigger.inputPathName trigger.inputFileName ] ) ;
152         triggerData = triggerData( : , triggerDataList.channel( cnt ) );
153         if fs ~= 44100 % Resample if needed
154             triggerData = resample( triggerData, 44100, fs );
155         end
156
157         % Trim data based on value in the TriggerThreshold column of the triggerDataList
158         %(comes from handles.data)
159         thresholdPercent = triggerDataList.TriggerThreshold ( cnt ) ;
160         triggerDataScaledTo100 = ( triggerData/max( triggerData ) ) * 100;
161         idx_Trim = find( triggerDataScaledTo100 > cell2mat(thresholdPercent), 1 );
162
163         %Add a bit past the passby for filter transients. The buffer gets
164         %removed in Max_SLM_A_Fast_44p1_kHz. Note this is not really needed
165         %for the 2dB test, since we are just using the overall but is kept
166         %in for consistency witht he one-third OB calcs in the compliance
167         %test.
168
169         endBuffer = round( 0.1 * fs );
170         idx_Trim = min( [ length( driverData ) idx_Trim + endBuffer ] );
171
172         %Trim driver and passenger signals for passbys
173         driverData = driverData( 1 : idx_Trim ) ;
174         passengerData = passengerData( 1 : idx_Trim ) ;
175
176     end
177
178     % Trim the Stationary signal to 10 sec (10 * fs) per final rule section 7.1.1(c)
179     if strcmp( handles.opCondForLoop, 'Stationary' ) ;
180         if length ( driverData ) < 10 * fs
181             errorTimeofSignal = 'The Stationary Signal must be at least 10 seconds long' ;
182             disp 'The Stationary Signal must be at least 10 seconds long'
183         else
184             driverData = driverData( 1 : 10 * fs );
185             passengerData = passengerData( 1 : 10 * fs );
186             frontData = frontData( 1 : 10 * fs );
187         end
188     end
```

```
189
190         % Trim the Reverse signal to 10 sec (10 * fs) per final rule section 7.1.1(c)
191         if strcmp( handles.opCondForLoop, 'Reverse' ) ;
192             if length ( driverData ) < 10 * fs
193                 errorTimeofSignal = 'The Reverse Signal must be at least 10 seconds long' ;
194                 disp 'The Reverse Signal must be at least 10 seconds long'
195             else
196                 driverData = driverData( 1 : 10 * fs );
197                 passengerData = passengerData( 1 : 10 * fs );
198             end
199         end
200
201          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
202
203         % Process Vehicle Signal for Driver Side
204         disp( '     Computing SPL for Vehicle - Driver Side.' )
205         [ driverSlmResults ] = Max_SLM_A_Fast_44p1_kHz( driverData );
206         driver.vehicleOverallLevels_dBA( cnt )  = driverSlmResults.maxSPL;
207         driver.vehicleFOBLevels_dBA( cnt, : ) = driverSlmResults.oneThirdOctaveBandLevelsAtMax;
208         driver.inputFileNames( cnt ) = { driver.inputFileName } ;
209
210         % Process Pre-Ambient Signal for Driver Side
211         disp( '     Computing SPL for Pre-Ambient - Driver Side.' )
212         [ preAmbDriverSlmResults] = Min_SLM_A_Fast_44p1_kHz( preAmbDriverData );
213
214         % Process Post-Ambient Signal for Driver Side
215         disp( '     Computing SPL for Post-Ambient - Driver Side.' )
216         [ postAmbDriverslmResults ] = Min_SLM_A_Fast_44p1_kHz( postAmbDriverData );
217
218         %--- Pick the ambient measurement with the lower minimum level
219         if preAmbDriverSlmResults.minSPL < postAmbDriverslmResults.minSPL
220             driver.ambientOverallLevels_dBA( cnt ) = preAmbDriverSlmResults.minSPL;
221             driver.ambientFOBLevels_dBA( cnt, : ) = preAmbDriverSlmResults.oneThirdOctaveBandLevelsAtMin;
222             driver.ambientFileNames( cnt ) = { preAmbDriver.inputFileName };
223             driver.ambientLowerSide( cnt ) = { '(Pre-Ambient)' };
224
225         else
226             driver.ambientOverallLevels_dBA( cnt ) = postAmbDriverslmResults.minSPL;
227             driver.ambientFOBLevels_dBA( cnt, : ) = postAmbDriverslmResults.oneThirdOctaveBandLevelsAtMin;
228             driver.ambientFileNames( cnt ) = { postAmbDriver.inputFileName } ;
229             driver.ambientLowerSide( cnt ) = { '(Post-Ambient)' };
230         end
231
232         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
233
234         % Process Vehicle Signal for Passenger Side
235         disp( '     Computing SPL for Vehicle - Passenger Side.' )
236         [ passengerSlmResults ] = Max_SLM_A_Fast_44p1_kHz( passengerData );
237         passenger.vehicleOverallLevels_dBA( cnt ) = passengerSlmResults.maxSPL;
238         passenger.vehicleFOBLevels_dBA( cnt, : ) = passengerSlmResults.oneThirdOctaveBandLevelsAtMax;
```

```
239         passenger.inputFileNames( cnt ) = { passenger.inputFileName  } ;
240
241         % Process Pre-Ambient Signal for Passenger Side
242         disp( '     Computing SPL for Pre-Ambient - Passenger Side.' )
243         [ preAmbPassengerSlmResults ] = Min_SLM_A_Fast_44p1_kHz( preAmbPassengerData );
244
245         % Process Post-Ambient Signal for Passenger Side
246         disp( '     Computing SPL for Post-Ambient - Passenger Side.' )
247         [ postAmbPassengerslmResultsPost ] = Min_SLM_A_Fast_44p1_kHz( postAmbPassengerData );
248
249         %--- Pick the ambient measurement with the lower minimum level
250         if preAmbPassengerSlmResults.minSPL < postAmbPassengerslmResultsPost.minSPL
251             passenger.ambientOverallLevels_dBA( cnt ) = preAmbPassengerSlmResults.minSPL;
252             passenger.ambientFOBLevels_dBA( cnt, : ) = preAmbPassengerSlmResults.oneThirdOctaveBandLevelsAtMin;
253             passenger.ambientFileNames( cnt ) = { preAmbPassenger.inputFileName };
254             passenger.ambientLowerSide( cnt ) = { '(Pre-Ambient)' };
255
256         else
257             passenger.ambientOverallLevels_dBA( cnt ) = postAmbPassengerslmResultsPost.minSPL;
258             passenger.ambientFOBLevels_dBA( cnt, : ) = postAmbPassengerslmResultsPost.oneThirdOctaveBandLevelsAtMin;
259             passenger.ambientFileNames( cnt ) = { postAmbPassenger.inputFileName } ;
260             passenger.ambientLowerSide( cnt ) = { '(Post-Ambient)' };
261         end
262
263         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
264         % Select Stationary "Front" Files
265
266         if strcmp(handles.opCondForLoop, 'Stationary')
267             disp( '     Computing SPL for Vehicle - Front Side.' )
268             [ frontSlmResults ] = Max_SLM_A_Fast_44p1_kHz( frontData );
269             front.vehicleOverallLevels_dBA( cnt ) = frontSlmResults.maxSPL;
270             front.vehicleFOBLevels_dBA( cnt, : ) = frontSlmResults.oneThirdOctaveBandLevelsAtMax;
271             front.inputFileNames( cnt ) = { front.inputFileName  };
272
273             % Process Pre-Ambient Signal for Front Side
274             disp( '     Computing SPL for Pre-Ambient - Front Side.' )
275             [ preAmbFrontSlmResults] = Min_SLM_A_Fast_44p1_kHz( preAmbFrontData );
276
277             % Process Ambient Signals for Front Side
278             disp( '     Computing SPL for Post-Ambient.' )
279             [ postAmbFrontSlmResults ] = Min_SLM_A_Fast_44p1_kHz( postAmbFrontData );
280
281             %--- Pick the ambient measurement with the lower minimum level
282             if preAmbFrontSlmResults.minSPL < postAmbFrontSlmResults.minSPL
283                 front.ambientOverallLevels_dBA( cnt ) = preAmbFrontSlmResults.minSPL;
284                 front.ambientFOBLevels_dBA( cnt, : ) = preAmbFrontSlmResults.oneThirdOctaveBandLevelsAtMin;
285                 front.ambientFileNames( cnt ) = { preAmbFront.inputFileName };
286                 front.ambientLowerSide( cnt ) = { '(Pre-Ambient)' };
287
288             else
```

```
            front.ambientOverallLevels_dBA( cnt ) = postAmbFrontSlmResults.minSPL;
            front.ambientFOBLevels_dBA( cnt, : ) = postAmbFrontSlmResults.oneThirdOctaveBandLevelsAtMin;
            front.ambientFileNames( cnt ) = { postAmbFront.inputFileName } ;
            front.ambientLowerSide( cnt ) = { '(Post-Ambient)' };
        end

    else
        front.vehicleOverallLevels_dBA( cnt ) = -99;
        front.vehicleFOBLevels_dBA( cnt, : ) = -99 * ones( 29, 1 );
        front.ambientOverallLevels_dBA( cnt ) = -99;
        front.ambientFOBLevels_dBA( cnt, : ) = -99 * ones( 29, 1 );
        front.inputFileNames( cnt ) = { [] };
        front.ambientLowerSide( cnt ) = { [] };

        handles.frontSPL = front ;
    end

    handles.driverSPL = driver ;
    handles.passengerSPL = passenger ;
    handles.frontSPL = front ;

    cnt = cnt + 1;

end
```

```
function handles = dataImportBatch( handles )
% This function imports all data in the specified directory into the tables in
% GUI_fileLoad2.m based on a standardized file format(see
% getChannelMapBatchLoad.m and getFileNameFormat. After the files from
% the folder have been imported, users can import additional data
% individually or submit the imported data for use in the main GUI.

%% Get all file names from the directory
batchFiles = handles.batchFileNames; %all files from folder structure starting with parent folder, including all
operating conditions
numberBatchFiles = length( batchFiles );
batchSignalOperationNamesOnly = handles.batchSignalNamesOnly ; %operating conditions associated with each signal file
numberBatchSignalsOnly = handles.numberBatchSignalsOnly ; %number of signals in the batch directory (excludes ambients)
batchSignalFileNamesOnly = handles.batchSignalFileNamesOnly ;% file names for each signal in batch directory
(excludes ambients)
batchSignalTrialNumber = handles.batchSignalTrialNumber ;% trial number inferred from file name

%% Variable assignments based on File Format

switch handles.fileFormat
    case 'Version 1' %Currently only one file format version, but the switch has been created in anticipation of
    future versions
        %Channel mappings from getChannelMapBatchLoad.m
        batchDriverChannel = handles.batchDriverChannel ;
        batchPassengerChannel= handles.batchPassengerChannel ;
        batchDriverPreAmbientChannel = handles.batchDriverPreAmbientChannel ;
        batchDriverPostAmbientChannel = handles.batchDriverPostAmbientChannel ;
        batchPassengerPreAmbientChannel = handles.batchPassengerPreAmbientChannel ;
        batchPassengerPostAmbientChannel = handles.batchPassengerPostAmbientChannel ;

        % Variable assignments based on optype
        for ink = 1 : numberBatchSignalsOnly

            tempFileName = batchSignalFileNamesOnly( ink ) ;

            if strcmp( batchSignalOperationNamesOnly( ink ), '10Passby')
                sampleID = handles.batchSignalTrialNumber( ink ) ; %Trial number from file name assigned to sample ID
                batchDriverSignalFileName = tempFileName ;
                batchPassengerSignalFileName = tempFileName ;
                batchTriggerSignalFileName = tempFileName ;
                batchPreAmbientFileName = handles.fileNamePreAmbient10 ;
                batchPostAmbientFileName = handles.fileNamePostAmbient10 ;
                batchPathName = handles.path10kmh ; %Path for the operating condition
                batchOpCond= '10 km/h' ;

                %Cell variable for table display
                dat10kmh( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName,
                batchPostAmbientFileName, ...
                    batchPassengerSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName, ...
                    batchTriggerSignalFileName ];
```

```
47
48              end
49
50              if strcmp( batchSignalOperationNamesOnly( ink ), '20Passby')
51                  sampleID = handles.batchSignalTrialNumber( ink ) ;
52                  batchDriverSignalFileName = tempFileName ;
53                  batchPassengerSignalFileName = tempFileName ;
54                  batchTriggerSignalFileName = tempFileName ;
55                  batchPreAmbientFileName = handles.fileNamePreAmbient20 ;
56                  batchPostAmbientFileName = handles.fileNamePostAmbient20 ;
57                  batchPathName = handles.path20kmh ;
58                  batchOpCond = '20 km/h';
59
60                  %Cell variable for table display
61                  dat20kmh( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName,
                    batchPostAmbientFileName, ...
62                      batchPassengerSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName, ...
63                      batchTriggerSignalFileName ];
64              end
65
66              if strcmp( batchSignalOperationNamesOnly( ink ), '30Passby')
67                  sampleID = handles.batchSignalTrialNumber( ink ) ;
68                  batchDriverSignalFileName = tempFileName ;
69                  batchPassengerSignalFileName = tempFileName ;
70                  batchTriggerSignalFileName = tempFileName ;
71                  batchPreAmbientFileName = handles.fileNamePreAmbient30 ;
72                  batchPostAmbientFileName = handles.fileNamePostAmbient30 ;
73                  batchPathName = handles.path30kmh ;
74                  batchOpCond = '30 km/h' ;
75
76                  %Cell variable for table display
77                  dat30kmh( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName,
                    batchPostAmbientFileName, ...
78                      batchPassengerSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName, ...
79                      batchTriggerSignalFileName ];
80              end
81
82              if  strcmp( batchSignalOperationNamesOnly( ink ), 'Stationary')
83                  sampleID = handles.batchSignalTrialNumber( ink ) ;
84                  batchDriverSignalFileName = tempFileName ;
85                  batchPassengerSignalFileName = tempFileName ;
86                  batchFrontSignalFileName = tempFileName ;
87                  batchPreAmbientFileName = handles.fileNamePreAmbientStat ;
88                  batchPostAmbientFileName = handles.fileNamePostAmbientStat ;
89                  batchPathName = handles.pathStationary ;
90                  batchOpCond = 'Stationary' ;
91
92                  %Cell variable for table display
93                  datStationary( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName ,
                    batchPostAmbientFileName,...
```

```
                batchPassengerSignalFileName , batchPreAmbientFileName,batchPostAmbientFileName, ...
                batchFrontSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName ];

        end

        if  strcmp( batchSignalOperationNamesOnly( ink ), 'Reverse')
            sampleID = handles.batchSignalTrialNumber( ink ) ;
            batchDriverSignalFileName = tempFileName ;
            batchPassengerSignalFileName = tempFileName ;
            batchPreAmbientFileName = handles.fileNamePreAmbientRev ;
            batchPostAmbientFileName = handles.fileNamePostAmbientRev ;
            batchPathName = handles.pathReverse ;
            batchOpCond = 'Reverse' ;

            %Cell variable for table display
            datReverse( ink, : ) = [ sampleID , batchDriverSignalFileName, batchPreAmbientFileName,
            batchPostAmbientFileName,...
                batchPassengerSignalFileName , batchPreAmbientFileName, batchPostAmbientFileName ];
        end

        % Import data into handles for any operating condition
        %(if statement used for consisitency, check probably not needed)
        if strcmp( batchSignalOperationNamesOnly( ink ), '10Passby') || strcmp( batchSignalOperationNamesOnly( ink
         ), '20Passby') ||...
                strcmp( batchSignalOperationNamesOnly( ink ), '30Passby') || strcmp( batchSignalOperationNamesOnly
                ( ink ), 'Stationary') ||...
                strcmp( batchSignalOperationNamesOnly( ink ), 'Reverse')

            % Map variables for data import (driver/passenger)
            opCond = batchOpCond ;%defined in previous conditional statement within the loop
            sampleID = handles.batchSignalTrialNumber{ ink };

            %remapping variables for consistency with individual file load
            driverChannel = batchDriverChannel ;
            driverPreAmbientChannel = batchDriverPreAmbientChannel ;
            driverPostAmbientChannel = batchDriverPostAmbientChannel ;
            passengerChannel = batchPassengerChannel ;
            passengerPreAmbientChannel = batchPassengerPreAmbientChannel ;
            passengerPostAmbientChannel = batchPassengerPostAmbientChannel ;
            driverSignalFileName = char( batchDriverSignalFileName ) ;
            passengerSignalFileName = char( batchPassengerSignalFileName ) ;
            preAmbientFileName = batchPreAmbientFileName ;
            postAmbientFileName = batchPostAmbientFileName ;

            %Dataset variable holds all data for import
            dataImport = dataset( { { opCond; opCond; opCond; opCond; opCond; opCond }, 'opCond' }, ...
                { { 'driver'; 'driver'; 'driver'; 'passenger'; 'passenger'; 'passenger' }, 'micSide' }, ...
                { { sampleID; sampleID; sampleID; sampleID; sampleID; sampleID }, 'sampleID' }, ...
                { [ driverChannel; driverPreAmbientChannel; driverPostAmbientChannel; passengerChannel;
                passengerPreAmbientChannel; passengerPostAmbientChannel ], 'channel' }, ...
```

```
        { { 'signal'; 'preAmbient'; 'postAmbient'; 'signal'; 'preAmbient'; 'postAmbient' }, 'signalType'
        }, ...
        { { batchPathName; batchPathName; batchPathName; batchPathName; batchPathName; batchPathName },
        'pathName' }, ...
        { { driverSignalFileName; preAmbientFileName; postAmbientFileName; passengerSignalFileName;
        preAmbientFileName; postAmbientFileName }, 'fileName' }, ...
        { { []; []; []; []; []; [] }, 'TriggerThreshold' }, ...
        { { []; []; []; []; []; [] }, 'idx' } );

    %Casting to nominal format required for later code
    dataImport.opCond = nominal( dataImport.opCond );
    dataImport.micSide = nominal( dataImport.micSide );
    dataImport.sampleID = nominal( dataImport.sampleID );
    dataImport.signalType = nominal( dataImport.signalType );
    dataImport.pathName = nominal( dataImport.pathName );
    dataImport.fileName = nominal( dataImport.fileName );

    %Create/append row to aggregate data
    if isempty( handles.data )
        handles.data = dataImport;
    else
        handles.data = [ handles.data; dataImport ];
    end

    % Create trigger dataset for passbys
    if strcmp( batchSignalOperationNamesOnly( ink ), '10Passby' ) || strcmp( batchSignalOperationNamesOnly
    ( ink ), '20Passby' )|| strcmp( batchSignalOperationNamesOnly( ink ), '30Passby' )
        triggerChannel = handles.batchTriggerChannel ;
        triggerFileName = char( batchTriggerSignalFileName );%Defined in if statements above

        triggerImport = dataset( { { opCond }, 'opCond' }, ...
            { { 'all' }, 'micSide' }, ...
            { { sampleID }, 'sampleID' }, ...
            { triggerChannel, 'channel' }, ...
            { { 'trigger' }, 'signalType' }, ...
            { { batchPathName }, 'pathName' }, ...
            { { triggerFileName }, 'fileName' }, ...
            { { [20] }, 'TriggerThreshold' }, ...
            { { [] }, 'idx' } );

        triggerImport.opCond = nominal( triggerImport.opCond );
        triggerImport.micSide = nominal( triggerImport.micSide );
        triggerImport.sampleID = nominal( triggerImport.sampleID );
        triggerImport.signalType = nominal( triggerImport.signalType );
        triggerImport.pathName = nominal( triggerImport.pathName );
        triggerImport.fileName = nominal( triggerImport.fileName );

        handles.data = [ handles.data; triggerImport ];
    end
```

```
                %Create front dataset for stationary
                if strcmp( batchSignalOperationNamesOnly( ink ), 'Stationary' )

                    frontChannel = handles.batchFrontChannel ;
                    frontPreAmbientChannel = handles.batchFrontPreAmbientChannel;
                    frontPostAmbientChannel = handles.batchFrontPostAmbientChannel;
                    frontSignalFileName = char( batchFrontSignalFileName );

                    frontImport = dataset( { { opCond; opCond; opCond }, 'opCond' }, ...
                        { { 'front'; 'front'; 'front' }, 'micSide' }, ...
                        { { sampleID; sampleID; sampleID }, 'sampleID' }, ...
                        { [ frontChannel; frontPreAmbientChannel; frontPostAmbientChannel ], 'channel' }, ...
                        { { 'signal'; 'preAmbient'; 'postAmbient' }, 'signalType' }, ...
                        { { batchPathName; batchPathName; batchPathName }, 'pathName' }, ...
                        { { frontSignalFileName; preAmbientFileName; postAmbientFileName }, 'fileName' }, ...
                        { { []; []; [] }, 'TriggerThreshold' }, ...
                        { { []; []; [] }, 'idx' } );

                    frontImport.opCond = nominal( frontImport.opCond );
                    frontImport.micSide = nominal( frontImport.micSide );
                    frontImport.sampleID = nominal( frontImport.sampleID );
                    frontImport.signalType = nominal( frontImport.signalType );
                    frontImport.pathName = nominal( frontImport.pathName );
                    frontImport.fileName = nominal( frontImport.fileName );

                    handles.data = [ handles.data; frontImport ];
                end
            end
        end
    end

    %% Delete empty rows for table display
    datStationary(all(cellfun('isempty',datStationary),2),:) = [];
    dat10kmh(all(cellfun('isempty',dat10kmh),2),:) = [];
    dat20kmh(all(cellfun('isempty',dat20kmh),2),:) = [];
    dat30kmh(all(cellfun('isempty',dat30kmh),2),:) = [];
    datReverse(all(cellfun('isempty',datReverse),2),:) = [];

    %% Update handles
    handles.datStationary = datStationary ;
    handles.dat10kmh = dat10kmh ;
    handles.dat20kmh = dat20kmh ;
    handles.dat30kmh = dat30kmh ;
    handles.datReverse = datReverse ;

    setappdata( 0, 'data', handles.data);
    end
```

```m
function [ Hd, h ] = designFractionalOctaveBandFilters

% Use filterbuilder for design
% See also FDATool, FVtool, sosfilt, sos2tf, dfilt/filter

B  = 3;       % Bands per octave
N  = 6;       % Order
Fs = 48000;   % Sampling Frequency -- Use 48 kHz, yields larger range of stable filters

% Center frequency
% F0 = [ 25.12 31.62 39.81 50.12 63.1 79.43 100 125.89 158.49 199.53 ...
% 251.19 316.23 398.11 501.19 630.96 794.33 1000 1258.93 1584.89 1995.26 ...
% 2511.89 3162.28 3981.07 5011.87 6309.57 7943.28 10000 12589.25 15848.93 ...
% 19952.62 ];

F0 = [ 316.23 398.11 501.19 1995.26 2511.89 3162.28 3981.07 5011.87 ];


warning('off','signal:fspecs:octavewordncenterfreq:thisvalidate:InvalidF0')
for ink = 1:length(F0)
    h(ink) = fdesign.octave(B, 'Class 0', 'N,F0', N, F0(ink), Fs);
    Hd(ink) = design(h(ink), 'butter', 'SOSScaleNorm', 'Linf');
end
warning('on','signal:fspecs:octavewordncenterfreq:thisvalidate:InvalidF0')

doPlot = 'n';
if strcmpi(doPlot(1), 'y')
    for ink = 1:length(F0)
        [ H, W ] = freqz(Hd(ink),16384);
        semilogx(W, 10*log10(abs(H)))
        hold on
    end
    xlabel('Normalized Frequency')
    ylabel('Magnitude of Filter Response')
    axis([ 0.0001   10.0000 -50 10])
    grid
end
```

```
function [ directivityPass, dPass, dPass2 ] = directivityTest_avg( centralMicPerformance, centralMicPerformance2 )

%Reviewing 4-band test performance for at least one passing combination
Pre = sum( centralMicPerformance, 1 );

Pre( Pre < 1 ) = 0;
Pre( Pre >= 1 ) = 1;

dPass = sum( Pre );

if dPass > 0
    Pass = 2;
else
    Pass = 0;
end

%Reviewing 2-band test performance for at least one passing combination
Pre2 = sum( centralMicPerformance2, 1 );

Pre2( Pre2 < 1 ) = 0;
Pre2( Pre2 >= 1 ) = 1;

dPass2 = sum( Pre2 );

if dPass2 > 0
    Pass2 = 1;
else
    Pass2 = 0;
end

directivityPass = Pass + Pass2;
```

```
clear
clc

load FOBFIlters

HdOrig = Hd ;

fc = [315 , 400 , 500, 630 , 800 , 1000 , 1250 , 1600 , 2000 , 2500 , 3150 , 4000 , 5000 ] ;

for ink = 1 : length( fc )
idx( ink ) = knnsearch( F0', fc( ink ) ) ;

end

Hd = Hd( idx ) ;
F0 = F0( idx ) ;

save FOBFiltersNHTSA F0 Hd


```

```m
function handles = getChannelMapBatchLoad( handles )
% This function describes the expected relationship between channels and
% signals

%% Specify Version
% Written to anticipate additional versions of channel organization.
% Currently hard coded for just Version 1. Can add additional cases in the
% future if additional formats are needed.

% Version 1
% It is expected that .wav files include the following data. All operation
% types(stationary, passbys, and reverse) require a signal file (that
% includes driver and passenger signals), a pre-abmient, and a
% post-ambient. Passbys also require a trigger signal that is included in
% the driver/passenger signal file. Stationary ops also require a front
% signal that is included in the driver/passenger signal file.
%
% Stationary, Passbys, Reverse Signals:
%   Channel 1: Driver Signal
%   Channel 2: Passenger Signal
% Stationary:
%   Channel 3 (of operation signal file): Front Signal
% Passbys:
%   Channel 3 (of operation signal file): Trigger Signal
% Ambients:
%   Channel 1: Ambient Signal

channelMap = 'Version 1' ;

%% Variable assignments based on channel mapping for batch load

switch channelMap
    case 'Version 1'

        % Driver and Passenger Channel Numbers
        handles.batchDriverChannel = 1 ;
        handles.batchPassengerChannel = 2 ;

        % Trigger Channel Number
        handles.batchTriggerChannel = 3;

        % Front Channel Number
        handles.batchFrontChannel = 3;

        % Ambient Channel Numbers
        handles.batchDriverPreAmbientChannel = 1 ;
        handles.batchDriverPostAmbientChannel = 1 ;
        handles.batchPassengerPreAmbientChannel = 2 ;
        handles.batchPassengerPostAmbientChannel = 2 ;
        handles.batchFrontPreAmbientChannel = 3 ;
```

```
        handles.batchFrontPostAmbientChannel = 3 ;


end


```

```
% Create a list of component combinations

function idx = getCombinations_AtLeast( n, N, dn, span )

% n = number of discrete signal components desired, e.g. 4
% N = number of contiguous bands where the n components may be, e.g. 13
% dn = minimum bands between components, e.g. 1
% span = the number of bands that the components should span, e.g. 9

% 4 components in 13 bands with a min of 1 band spacing and a span of 9
% bands will produce 185 combinations

v = 1 : N; % number of bands
idx = nchoosek( v, n ); % all possible combinations of 4 bands
for ink = 1 : n - 1
    deltas = diff( idx' )'; % difference between bands
    idx = idx( deltas( :, ink ) >= dn + 1, : ); % remove pairs with a difference of 1 between bands
end

deltas = idx( :, end ) - idx( :, 1 ); % difference between last band and first band
idx = idx( deltas( :, 1 ) >= span - 1, : ); % removing groups that don't meet the min span

```

```
% Create a list of component combinations

function idx = getCombinations_band2( n, N, dn, fc, lowFreqCriterion, span )

% n = number of discrete signal components desired, e.g. 4
% N = number of contiguous bands where the n components may be, e.g. 13
% dn = minimum bands between components, e.g. 1
% span = the number of bands that the components should span, e.g. 9
%
% 4 components in 13 bands with a min of 1 band spacing and a span of 9
% bands will produce 185 combinations

v = 1 : N; % number of bands
idx = nchoosek( v, n ); % all possible combinations of pairs of bands (2 band test)

for ink = 1 : n - 1
    deltas = diff( idx' )'; % difference between bands in the pair
    idx = idx( deltas( :, ink ) >= dn + 1, : ); % remove pairs with a difference of 1 between bands
end

%find(X,K,'last') returns at most the last K indices corresponding to the nonzero entries of the array X.
%finding the last freq equal to or below 1000Hz
idxCrit = find( fc <= lowFreqCriterion, 1, 'last' );

idx = idx( idx( :, 1 ) < idxCrit, : ); % first column of idx that is less than 1000 Hz (index 6)
idx = idx( idx( :, 2 ) >= idxCrit, : ); % second column if idx that is greater than or equal to 1000 Hz

deltas = idx( :, end ) - idx( :, 1 ); % difference between bands in the pair
idx = idx( deltas( :, 1 ) >= span - 1, : ); % removing pairs that don't meet the min span

```

```m
function handles = getFileNameFormat( handles )
% This function describes the expected file name format.

%% Specify File Format
% Written to anticipate additional versions of file formats.
% Currently hard coded for just Version 1. Can add additional cases in the future if additional formats are needed.

% Version 1 Format
% Example: VRTC_20170329_0507_Ford_Fusion_2010_ICE_30PassBy_002.wav
% VRTC: TestLocation
% 20170329: yyyymmdd
% 0507: 4-char daily sequence number
% Ford: Make
% Fusion: Model
% 2010: Model Year
% ICE: Vehicle type (ICE/EV/HEV)
% 30PassBy: Op type (30PassBy, 20PassBy, 10PassBy, Stationary, Reverse)
% 02: 3- digit trial number

fileFormat = 'Version 1' ;

%% Variable assignments based on file format

%Get all files names from the directory
batchFiles = handles.batchFileNames;
numberBatchFiles = length( batchFiles );

switch fileFormat
    case 'Version 1'

        % Variable Preallocation
        batchTestLocation = cell ( 1, numberBatchFiles ) ;
        batchDate = cell ( 1, numberBatchFiles ) ;
        batchSeqNumber = cell ( 1, numberBatchFiles ) ;
        batchMake = cell ( 1, numberBatchFiles ) ;
        batchModel = cell ( 1, numberBatchFiles ) ;
        batchModelYear = cell ( 1, numberBatchFiles ) ;
        batchVehicleType = cell ( 1, numberBatchFiles ) ;
        batchOpType = cell ( 1, numberBatchFiles ) ;
        batchTrialNumber = cell ( 1, numberBatchFiles ) ;

        for ink = 1 : numberBatchFiles
            tempFileName = batchFiles( ink ).name ;
            underscoreLocations = strfind( tempFileName, '_') ;

            % File Name Based Parameters
            batchTestLocation{ ink } = tempFileName( 1  : underscoreLocations( 1 ) - 1 ); % (VRTC in Version 1
            example above)
            batchDate{ ink } = tempFileName( underscoreLocations( 1 ) + 1  : underscoreLocations( 2 ) - 1 ); %
            (20170329 in Version 1 example above)
```

```
            batchSeqNumber{ ink } = tempFileName( underscoreLocations( 2 ) + 1  : underscoreLocations( 3 ) - 1 ); %
            (0507 in Version 1 example above)
            batchMake{ ink } = tempFileName( underscoreLocations( 3 ) + 1  : underscoreLocations( 4 ) - 1 ); % (Ford
            in Version 1 example above)
            batchModel{ ink } = tempFileName( underscoreLocations( 4 ) + 1  : underscoreLocations( 5 ) - 1 ); %
            (Fusion in Version 1 example above)
            batchModelYear{ ink } = tempFileName( underscoreLocations( 5 ) + 1  : underscoreLocations( 6 ) - 1 ); %
            (2010 in Version 1 example above)
            batchVehicleType{ ink } = tempFileName( underscoreLocations( 6 ) + 1  : underscoreLocations( 7 ) - 1 ); %
            (ICE in Version 1 example above)
            batchOpType{ ink } = tempFileName( underscoreLocations( 7 ) + 1  : underscoreLocations( 8 ) - 1 ); %
            (30PassBy in Version 1 example above)
            batchTrialNumber{ ink } = tempFileName( underscoreLocations( 8 ) + 1 : end - 4 ); % (002 in Version 1
            example above)

        end

        % Number of signal files only
        idxAmbients = strfind( batchOpType, 'Ambient' ) ;
        removeAmbients = cellfun( 'isempty', idxAmbients) ;
        batchSignalNamesOnly = batchOpType( removeAmbients ) ;
        numberBatchSignalsOnly = length( batchSignalNamesOnly ) ;
        batchSignalFileNamesOnly = { batchFiles( removeAmbients ).name };
        handles.batchSignalTrialNumber = batchTrialNumber( removeAmbients );

        % Ambient File Names
        idxPreAmbient10 = strfind( batchOpType, '10PreAmbient' ) ;
        retainOnlyPreAmbient10 = find( ~cellfun( @isempty, idxPreAmbient10 )) ;
        PreAmbient10 = batchFiles( retainOnlyPreAmbient10 ) ;

        idxPostAmbient10 = strfind( batchOpType, '10PostAmbient' ) ;
        retainOnlyPostAmbient10 = find( ~cellfun( @isempty, idxPostAmbient10 )) ;
        PostAmbient10 = batchFiles( retainOnlyPostAmbient10 ) ;

        idxPreAmbient20 = strfind( batchOpType, '20PreAmbient' ) ;
        retainOnlyPreAmbient20 = find( ~cellfun( @isempty, idxPreAmbient20 )) ;
        PreAmbient20 = batchFiles( retainOnlyPreAmbient20 ) ;

        idxPostAmbient20 = strfind( batchOpType, '20PostAmbient' ) ;
        retainOnlyPostAmbient20 = find( ~cellfun( @isempty, idxPostAmbient20 )) ;
        PostAmbient20 = batchFiles( retainOnlyPostAmbient20 ) ;

        idxPreAmbient30 = strfind( batchOpType, '30PreAmbient' ) ;
        retainOnlyPreAmbient30 = find( ~cellfun( @isempty, idxPreAmbient30 )) ;
        PreAmbient30 = batchFiles( retainOnlyPreAmbient30 ) ;

        idxPostAmbient30 = strfind( batchOpType, '30PostAmbient' ) ;
        retainOnlyPostAmbient30 = find( ~cellfun( @isempty, idxPostAmbient30 )) ;
        PostAmbient30 = batchFiles( retainOnlyPostAmbient30 ) ;

```

```
        idxPreAmbientStat = strfind( batchOpType, 'StatPreAmbient' ) ;
        retainOnlyPreAmbientStat = find( ~cellfun( @isempty, idxPreAmbientStat )) ;
        PreAmbientStat = batchFiles( retainOnlyPreAmbientStat ) ;

        idxPostAmbientStat = strfind( batchOpType, 'StatPostAmbient' ) ;
        retainOnlyPostAmbientStat = find( ~cellfun( @isempty, idxPostAmbientStat )) ;
        PostAmbientStat = batchFiles( retainOnlyPostAmbientStat ) ;

        idxPreAmbientRev = strfind( batchOpType, 'RevPreAmbient' ) ;
        retainOnlyPreAmbientRev = find( ~cellfun( @isempty, idxPreAmbientRev )) ;
        PreAmbientRev = batchFiles( retainOnlyPreAmbientRev ) ;

        idxPostAmbientRev = strfind( batchOpType, 'RevPostAmbient' ) ;
        retainOnlyPostAmbientRev = find( ~cellfun( @isempty, idxPostAmbientRev )) ;
        PostAmbientRev = batchFiles( retainOnlyPostAmbientRev ) ;

        % Update remaining handles
        handles.batchTestLocation = batchTestLocation ;
        handles.batchDate = batchDate ;
        handles.batchSeqNumber = batchSeqNumber ;
        handles.batchMake = batchMake ;
        handles.batchModel = batchModel ;
        handles.batchModelYear = batchModelYear ;
        handles.batchVehicleType = batchVehicleType ;
        handles.batchOpType = batchOpType ;
        handles.batchTrialNumber = batchTrialNumber ;
        handles.batchSignalNamesOnly = batchSignalNamesOnly ;
        handles.numberBatchSignalsOnly = numberBatchSignalsOnly ;
        handles.batchSignalFileNamesOnly = batchSignalFileNamesOnly;
        handles.fileFormat = fileFormat;

        handles.fileNamePreAmbient10 = PreAmbient10.name ;
        handles.fileNamePostAmbient10 = PostAmbient10.name ;
        handles.fileNamePreAmbient20 = PreAmbient20.name ;
        handles.fileNamePostAmbient20 = PostAmbient20.name ;
        handles.fileNamePreAmbient30 = PreAmbient30.name ;
        handles.fileNamePostAmbient30 = PostAmbient30.name ;
        handles.fileNamePreAmbientStat = PreAmbientStat.name ;
        handles.fileNamePostAmbientStat = PostAmbientStat.name ;
        handles.fileNamePreAmbientRev = PreAmbientRev.name ;
        handles.fileNamePostAmbientRev = PostAmbientRev.name ;

end
```

```
1
2   function varargout = GUI_fileLoad2(varargin)
3   % GUI_fileLoad2 MATLAB code for GUI_fileLoad2.fig
4   %      GUI_fileLoad2, by itself, creates a new GUI_fileLoad2 or raises the existing
5   %      singleton*.
6   %
7   %      H = GUI_fileLoad2 returns the handle to a new GUI_fileLoad2 or the handle to
8   %      the existing singleton*.
9   %
10  %      GUI_fileLoad2('CALLBACK',hObject,eventData,handles,...) calls the local
11  %      function named CALLBACK in GUI_fileLoad2.M with the given input arguments.
12  %
13  %      GUI_fileLoad2('Property','Value',...) creates a new GUI_fileLoad2 or raises the
14  %      existing singleton*.  Starting from the left, property value pairs are
15  %      applied to the GUI before GUI_fileLoad2_OpeningFcn gets called.  An
16  %      unrecognized property name or invalid value makes property application
17  %      stop.  All inputs are passed to GUI_fileLoad2_OpeningFcn via varargin.
18  %
19  %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
20  %      instance to run (singleton)".
21  %
22  % See also: GUIDE, GUIDATA, GUIHANDLES
23
24  % Edit the above text to modify the response to help GUI_fileLoad2
25
26  % Last Modified by GUIDE v2.5 01-Aug-2018 13:22:55
27
28  %**************************************************************************
29  %**************************************************************************
30  %       Creation
31  %**************************************************************************
32  %**************************************************************************
33
34  %--------------------------------------------------------------------------
35  % Begin initialization code - DO NOT EDIT
36  gui_Singleton = 1;
37  gui_State = struct('gui_Name',       mfilename, ...
38      'gui_Singleton',  gui_Singleton, ...
39      'gui_OpeningFcn', @GUI_fileLoad2_OpeningFcn, ...
40      'gui_OutputFcn',  @GUI_fileLoad2_OutputFcn, ...
41      'gui_LayoutFcn',  [] , ...
42      'gui_Callback',   []);
43  if nargin && ischar(varargin{1})
44      gui_State.gui_Callback = str2func(varargin{1});
45  end
46
47  if nargout
48      [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
49  else
50      gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT

% --- Executes just before GUI_fileLoad2 is made visible.
function GUI_fileLoad2_OpeningFcn(hObject, ~, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_fileLoad2 (see VARARGIN)

% Clear the console display
clc

% Set the name of the GUI window
set(handles.figure1, 'Name', 'NHTSA Quiet Vehicle Compliance Tool - File Load');

%Assign variables based on batch file names
handles.output = hObject;
handles.fileLoadCondition = getappdata( 0, 'fileLoadCondition' ) ;
handles.datStationary  = getappdata( 0, 'datStationary' ) ;
handles.dat10kmh = getappdata( 0, 'dat10kmh' ) ;
handles.dat20kmh  = getappdata( 0, 'dat20kmh') ;
handles.dat30kmh = getappdata( 0, 'dat30kmh' ) ;
handles.datReverse =  getappdata( 0, 'datReverse'  ) ;

handles.data = getappdata( 0, 'data' );

handles.useTestData = 0 ; % Flag for testing purposes. 1 to use test data, 0 to enter data

%Table set up
set( handles.tableStationary, 'ColumnFormat', { 'char' , 'char', 'char', 'char', 'char', 'char', 'char'},'ColumnWidth'
, { 75 130 150 150 150 165 165 130 150 150 } );
set( handles.table10kmh, 'ColumnFormat', { 'char' , 'char', 'char', 'char', 'char', 'char' },'ColumnWidth', { 75 130
150 150 150 165 165 80 } );
set( handles.table20kmh, 'ColumnFormat', { 'char' , 'char', 'char', 'char', 'char', 'char' },'ColumnWidth', { 75 130
150 150 150 165 165 80 } );
set( handles.table30kmh, 'ColumnFormat', { 'char' , 'char', 'char', 'char', 'char', 'char' },'ColumnWidth', { 75 130
150 150 150 165 165 80 } );
set( handles.tableReverse, 'ColumnFormat', { 'char' , 'char', 'char', 'char', 'char'},'ColumnWidth', { 75 130 150 150
150 165 165 } );

opC = { 'Stationary', '10 km/h', '20 km/h', '30 km/h', 'Reverse' } ;
tab = { 'tableStationary', 'table10kmh', 'table20kmh', 'table30kmh', 'tableReverse' };

try % Look for flag that indicates if data has already been imported
    handles.loadedTableDataFlag = getappdata( 0, 'loadedTableDataFlag')  ;
catch
    handles.loadedTableDataFlag = 0 ;
end
```

```
96
97  if handles.loadedTableDataFlag % Load data into the tables if data exists
98      if handles.useTestData
99          if ~isempty( handles.data )
100             for ink = 1 : length( opC )
101                 handles = displayTable( handles, opC{ ink }, tab{ ink } );
102             end
103         end
104     else
105         handles.loadedTableDataStationary = getappdata( 0, 'loadedTableDataStationary' );
106         handles.loadedTableData10kmh = getappdata( 0, 'loadedTableData10kmh' );
107         handles.loadedTableData20kmh = getappdata( 0, 'loadedTableData20kmh' );
108         handles.loadedTableData30kmh = getappdata( 0, 'loadedTableData30kmh' );
109         handles.loadedTableDataReverse = getappdata( 0, 'loadedTableDataReverse' );
110
111         set( handles.tableStationary, 'Data', handles.loadedTableDataStationary );
112         set( handles.table10kmh, 'Data', handles.loadedTableData10kmh );
113         set( handles.table20kmh, 'Data', handles.loadedTableData20kmh );
114         set( handles.table30kmh, 'Data', handles.loadedTableData30kmh );
115         set( handles.tableReverse, 'Data', handles.loadedTableDataReverse );
116     end
117
118     % Enable the submit data button if at least for samples for each operating condition
119     if size (handles.loadedTableDataStationary, 1 ) >= 4 && size (handles.loadedTableData10kmh , 1 ) >= 4 && size (
        handles.loadedTableData20kmh , 1 ) >= 4 ...
120             && size (handles.loadedTableData30kmh  , 1 ) >= 4 && size (handles.loadedTableDataReverse , 1 ) >= 4
121         set( handles.buttonSubmitData, 'Enable', 'on' );
122         set( handles.textDataSubmittalMessage, 'String', ' ' );
123     else
124         set( handles.buttonSubmitData, 'Enable', 'off' );
125     end
126 end
127
128 if isfield( handles, 'fileLoadCondition' )
129     if strcmp( handles.fileLoadCondition, 'batch' )
130         handles = displayTable( handles ); % Display batch loaded data in tables if importing by batch
131     end
132 end
133
134 % Update handles structure
135 guidata(hObject, handles);
136
137 %**************************************************************************
138 %**************************************************************************
139 %       Callbacks and Button Presses
140 %**************************************************************************
141 %**************************************************************************
142
143 % --- Executes on selection change in OperatingConditionFileLoadMenu.
144 function OperatingConditionFileLoadMenu_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to OperatingConditionFileLoadMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns OperatingConditionFileLoadMenu contents as cell array
%        contents{get(hObject,'Value')} returns selected item from OperatingConditionFileLoadMenu

contents = cellstr( get( hObject, 'String' ) );
handles.operatingCondition = contents{ get( hObject, 'Value' ) };

% Enable fields based on operating condition selection
switch handles.operatingCondition
    case 'Stationary'
        handles.opCond = 'Stationary';
        set(handles.textTriggerTitle, 'enable', 'off')
        set(handles.textTrigger, 'enable', 'off')
        set(handles.pushbuttonSelectTrigger, 'enable', 'off')
        set(handles.controlTriggerChannel, 'enable', 'off')

        set( handles.pushbuttonSelectFrontSignal, 'Enable', 'on' );
        set( handles.controlFrontSignalChannel, 'Enable', 'on' );
        set( handles.pushbuttonSelectFrontPreAmbient, 'Enable', 'on' );
        set( handles.controlFrontPreAmbientChannel, 'Enable', 'on' );
        set( handles.pushbuttonSelectFrontPostAmbient, 'Enable', 'on' );
        set( handles.controlFrontPostAmbientChannel, 'Enable', 'on' );
        set( handles.textFront, 'Enable', 'on' );
        set( handles.textFrontSignal, 'Enable', 'on' );
        set( handles.textFrontPreAmbient, 'Enable', 'on' );
        set( handles.textFrontPostAmbient, 'Enable', 'on' );

    case '10 km/h'
        handles.opCond = '10 km/h';
        set(handles.textTriggerTitle, 'enable', 'on')
        set(handles.textTrigger, 'enable', 'on')
        set(handles.pushbuttonSelectTrigger, 'enable', 'on')
        set(handles.controlTriggerChannel, 'enable', 'on')

        set( handles.pushbuttonSelectFrontSignal, 'Enable', 'off' );
        set( handles.controlFrontSignalChannel, 'Enable', 'off' );
        set( handles.pushbuttonSelectFrontPreAmbient, 'Enable', 'off' );
        set( handles.controlFrontPreAmbientChannel, 'Enable', 'off' );
        set( handles.pushbuttonSelectFrontPostAmbient, 'Enable', 'off' );
        set( handles.controlFrontPostAmbientChannel, 'Enable', 'off' );
        set( handles.textFront, 'Enable', 'off' );
        set( handles.textFrontSignal, 'Enable', 'off' );
        set( handles.textFrontPreAmbient, 'Enable', 'off' );
        set( handles.textFrontPostAmbient, 'Enable', 'off' );

    case '20 km/h'
        handles.opCond = '20 km/h';
```

```
195             set(handles.textTriggerTitle, 'enable', 'on')
196             set(handles.textTrigger, 'enable', 'on')
197             set(handles.pushbuttonSelectTrigger, 'enable', 'on')
198             set(handles.controlTriggerChannel, 'enable', 'on')
199 
200             set( handles.pushbuttonSelectFrontSignal, 'Enable', 'off' );
201             set( handles.controlFrontSignalChannel, 'Enable', 'off' );
202             set( handles.pushbuttonSelectFrontPreAmbient, 'Enable', 'off' );
203             set( handles.controlFrontPreAmbientChannel, 'Enable', 'off' );
204             set( handles.pushbuttonSelectFrontPostAmbient, 'Enable', 'off' );
205             set( handles.controlFrontPostAmbientChannel, 'Enable', 'off' );
206             set( handles.textFront, 'Enable', 'off' );
207             set( handles.textFrontSignal, 'Enable', 'off' );
208             set( handles.textFrontPreAmbient, 'Enable', 'off' );
209             set( handles.textFrontPostAmbient, 'Enable', 'off' );
210 
211         case '30 km/h'
212             handles.opCond = '30 km/h';
213             set(handles.textTriggerTitle, 'enable', 'on')
214             set(handles.textTrigger, 'enable', 'on')
215             set(handles.pushbuttonSelectTrigger, 'enable', 'on')
216             set(handles.controlTriggerChannel, 'enable', 'on')
217 
218             set( handles.pushbuttonSelectFrontSignal, 'Enable', 'off' );
219             set( handles.controlFrontSignalChannel, 'Enable', 'off' );
220             set( handles.pushbuttonSelectFrontPreAmbient, 'Enable', 'off' );
221             set( handles.controlFrontPreAmbientChannel, 'Enable', 'off' );
222             set( handles.pushbuttonSelectFrontPostAmbient, 'Enable', 'off' );
223             set( handles.controlFrontPostAmbientChannel, 'Enable', 'off' );
224             set( handles.textFront, 'Enable', 'off' );
225             set( handles.textFrontSignal, 'Enable', 'off' );
226             set( handles.textFrontPreAmbient, 'Enable', 'off' );
227             set( handles.textFrontPostAmbient, 'Enable', 'off' );
228 
229         case 'Reverse'
230             handles.opCond = 'Reverse';
231             set(handles.textTriggerTitle, 'enable', 'off')
232             set(handles.textTrigger, 'enable', 'off')
233             set(handles.pushbuttonSelectTrigger, 'enable', 'off')
234             set(handles.controlTriggerChannel, 'enable', 'off')
235 
236             set( handles.pushbuttonSelectFrontSignal, 'Enable', 'off' );
237             set( handles.controlFrontSignalChannel, 'Enable', 'off' );
238             set( handles.pushbuttonSelectFrontPreAmbient, 'Enable', 'off' );
239             set( handles.controlFrontPreAmbientChannel, 'Enable', 'off' );
240             set( handles.pushbuttonSelectFrontPostAmbient, 'Enable', 'off' );
241             set( handles.controlFrontPostAmbientChannel, 'Enable', 'off' );
242             set( handles.textFront, 'Enable', 'off' );
243             set( handles.textFrontSignal, 'Enable', 'off' );
244             set( handles.textFrontPreAmbient, 'Enable', 'off' );
```

```
        set( handles.textFrontPostAmbient, 'Enable', 'off' );
end

% Update handles structure
guidata( hObject, handles );

function editSampleID_Callback(hObject, ~, handles)
% hObject    handle to editSampleID (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editSampleID as text
%        str2double(get(hObject,'String')) returns contents of editSampleID as a double
handles.sampleID = get( hObject, 'String' );

guidata( hObject, handles );

% --- Executes on button press in pushbuttonSelectDriverSignal.
function pushbuttonSelectDriverSignal_Callback(hObject, ~, handles)

getFileFilter = '*.wav';

if isfield( handles, 'getFileFilter' )
    getFileFilter = handles.getFileFilter;
end

[ handles.driverSignalFileName, handles.driverSignalPathName ] = ...
    uigetfile( getFileFilter, 'Load **Signal** File' );

if handles.driverSignalFileName ~= 0
    set( handles.displayDriverSignalFile, 'String', [ handles.driverSignalPathName, '\', handles.driverSignalFileName
    ] );
    set( handles.displayDriverSignalFile, 'Enable', 'on' );
end

% Update handles structure
guidata( hObject, handles );

% --- Executes on selection change in controlDriverSignalChannel.
function controlDriverSignalChannel_Callback(hObject, ~, handles)
handles.driverChannel = get(hObject,'Value');

% Update handles structure
guidata( hObject, handles );

% --- Executes on button press in pushbuttonSelectDriverPreAmbient.
function pushbuttonSelectDriverPreAmbient_Callback(hObject, ~, handles)

getFileFilter = '*.wav';

```

```
294   if isfield( handles, 'getFileFilter' )
295       getFileFilter = handles.getFileFilter;
296   end
297
298   [ handles.driverPreAmbientFileName, handles.driverPreAmbientPathName ] = ...
299       uigetfile( getFileFilter, 'Load **Ambient** File' );
300
301   if handles.driverPreAmbientFileName ~= 0
302       set( handles.displayDriverPreAmbientFile, 'String', [ handles.driverPreAmbientPathName, '\', handles.
      driverPreAmbientFileName ] );
303       set( handles.displayDriverPreAmbientFile, 'Enable', 'on' );
304   end
305
306   % Update handles structure
307   guidata( hObject, handles );
308
309   % --- Executes on selection change in controlDriverPreAmbientChannel.
310   function controlDriverPreAmbientChannel_Callback(hObject, ~, handles)
311   handles.driverPreAmbientChannel = get( hObject, 'Value' );
312
313   % Update handles structure
314   guidata( hObject, handles );
315
316   % --- Executes on button press in pushbuttonSelectDriverPostAmbient.
317   function pushbuttonSelectDriverPostAmbient_Callback(hObject, eventdata, handles)
318   % hObject    handle to pushbuttonSelectDriverPostAmbient (see GCBO)
319   % eventdata  reserved - to be defined in a future version of MATLAB
320   % handles    structure with handles and user data (see GUIDATA)
321
322   getFileFilter = '*.wav';
323
324   if isfield( handles, 'getFileFilter' )
325       getFileFilter = handles.getFileFilter;
326   end
327
328   [ handles.driverPostAmbientFileName, handles.driverPostAmbientPathName ] = ...
329       uigetfile( getFileFilter, 'Load **Ambient** File' );
330
331   if handles.driverPostAmbientFileName ~= 0
332       set( handles.displayDriverPostAmbientFile, 'String', [ handles.driverPostAmbientPathName, '\', handles.
      driverPostAmbientFileName ] );
333       set( handles.displayDriverPostAmbientFile, 'Enable', 'on' );
334   end
335
336   % Update handles structure
337   guidata( hObject, handles );
338
339   % --- Executes on selection change in controlDriverPostAmbientChannel.
340   function controlDriverPostAmbientChannel_Callback(hObject, eventdata, handles)
341   handles.driverPostAmbientChannel = get( hObject, 'Value' );
```

```
342
343   % Update handles structure
344   guidata( hObject, handles );
345
346   % --- Executes on button press in pushbuttonSelectPassengerSignal.
347   function pushbuttonSelectPassengerSignal_Callback(hObject, ~, handles)
348
349   getFileFilter = '*.wav';
350
351   if isfield( handles, 'getFileFilter' )
352       getFileFilter = handles.getFileFilter;
353   end
354
355   [ handles.passengerSignalFileName, handles.passengerSignalPathName ] = ...
356       uigetfile( getFileFilter, 'Load **Signal** File' );
357
358   if handles.passengerSignalFileName ~= 0
359       set( handles.displayPassengerSignalFile, 'String', [ handles.passengerSignalPathName, '\', handles.
          passengerSignalFileName ] );
360       set( handles.displayPassengerSignalFile, 'Enable', 'on' );
361   end
362
363   % Update handles structure
364   guidata( hObject, handles );
365
366   % --- Executes on selection change in controlPassengerSignalChannel.
367   function controlPassengerSignalChannel_Callback(hObject, ~, handles)
368   handles.passengerChannel = get( hObject, 'Value' );
369
370   % Update handles structure
371   guidata( hObject, handles );
372
373   % --- Executes on button press in pushbuttonSelectPassengerPreAmbient.
374   function pushbuttonSelectPassengerPreAmbient_Callback(hObject, ~, handles)
375
376   getFileFilter = '*.wav';
377
378   if isfield( handles, 'getFileFilter' )
379       getFileFilter = handles.getFileFilter;
380   end
381
382   [ handles.passengerPreAmbientFileName, handles.passengerPreAmbientPathName ] = ...
383       uigetfile( getFileFilter, 'Load **Ambient** File' );
384
385   if handles.passengerPreAmbientFileName ~= 0
386       set( handles.displayPassengerPreAmbientFile, 'String', ...
387           [ handles.passengerPreAmbientPathName, '\', handles.passengerPreAmbientFileName ] );
388       set( handles.displayPassengerPreAmbientFile, 'Enable', 'on' );
389   end
390
```

```
391   % Update handles structure
392   guidata( hObject, handles );
393
394   % --- Executes on selection change in controlPassengerPreAmbientChannel.
395   function controlPassengerPreAmbientChannel_Callback(hObject, ~, handles)
396   handles.passengerPreAmbientChannel = get( hObject, 'Value' );
397
398   % Update handles structure
399   guidata( hObject, handles );
400
401   % --- Executes on button press in pushbuttonSelectPassengerPostAmbient.
402   function pushbuttonSelectPassengerPostAmbient_Callback(hObject, eventdata, handles)
403   % hObject    handle to pushbuttonSelectPassengerPostAmbient (see GCBO)
404   % eventdata  reserved - to be defined in a future version of MATLAB
405   % handles    structure with handles and user data (see GUIDATA)
406
407   getFileFilter = '*.wav';
408
409   if isfield( handles, 'getFileFilter' )
410       getFileFilter = handles.getFileFilter;
411   end
412
413   [ handles.passengerPostAmbientFileName, handles.passengerPostAmbientPathName ] = ...
414       uigetfile( getFileFilter, 'Load **Ambient** File' );
415
416   if handles.passengerPostAmbientFileName ~= 0
417       set( handles.displayPassengerPostAmbientFile, 'String', [ handles.passengerPostAmbientPathName, '\', handles.
      passengerPostAmbientFileName ] );
418       set( handles.displayPassengerPostAmbientFile, 'Enable', 'on' );
419   end
420
421   % Update handles structure
422   guidata( hObject, handles );
423
424   % --- Executes on selection change in controlPassengerPostAmbientChannel.
425   function controlPassengerPostAmbientChannel_Callback(hObject, eventdata, handles)
426   handles.passengerPostAmbientChannel = get( hObject, 'Value' );
427
428   % Update handles structure
429   guidata( hObject, handles );
430
431   % --- Executes on button press in pushbuttonSelectFrontSignal.
432   function pushbuttonSelectFrontSignal_Callback(hObject, ~, handles)
433
434   getFileFilter = '*.wav';
435
436   if isfield( handles, 'getFileFilter' )
437       getFileFilter = handles.getFileFilter;
438   end
439
```

```
[ handles.frontSignalFileName, handles.frontSignalPathName ] = ...
    uigetfile( getFileFilter, 'Load **Signal** File' );

if handles.frontSignalFileName ~= 0
    set( handles.displayFrontSignalFile, 'String', [ handles.frontSignalPathName, '\', handles.frontSignalFileName ] );
    set( handles.displayFrontSignalFile, 'Enable', 'on' );
end

% Update handles structure
guidata( hObject, handles );

% --- Executes on selection change in controlFrontSignalChannel.
function controlFrontSignalChannel_Callback(hObject, ~, handles)
handles.frontChannel = get( hObject, 'Value' );

% Update handles structure
guidata( hObject, handles );

% --- Executes on button press in pushbuttonSelectFrontPreAmbient.
function pushbuttonSelectFrontPreAmbient_Callback(hObject, ~, handles)

getFileFilter = '*.wav';

if isfield( handles, 'getFileFilter' )
    getFileFilter = handles.getFileFilter;
end

[ handles.frontPreAmbientFileName, handles.frontPreAmbientPathName ] = ...
    uigetfile( getFileFilter, 'Load **Ambient** File' );

if handles.frontPreAmbientFileName ~= 0
    set( handles.displayFrontPreAmbientFile, 'String', [ handles.frontPreAmbientPathName, '\', handles.
    frontPreAmbientFileName ] );
    set( handles.displayFrontPreAmbientFile, 'Enable', 'on' );
end

% Update handles structure
guidata( hObject, handles );

% --- Executes on selection change in controlFrontPreAmbientChannel.
function controlFrontPreAmbientChannel_Callback(hObject, ~, handles)
handles.frontPreAmbientChannel = get( hObject, 'Value' );

% Update handles structure
guidata( hObject, handles );

% --- Executes on button press in pushbuttonSelectFrontPostAmbient.
function pushbuttonSelectFrontPostAmbient_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonSelectFrontPostAmbient (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

getFileFilter = '*.wav';

if isfield( handles, 'getFileFilter' )
    getFileFilter = handles.getFileFilter;
end

[ handles.frontPostAmbientFileName, handles.frontPostAmbientPathName ] = ...
    uigetfile( getFileFilter, 'Load **Ambient** File' );

if handles.frontPostAmbientFileName ~= 0
    set( handles.displayFrontPostAmbientFile, 'String', [ handles.frontPostAmbientPathName, '\', handles.
    frontPostAmbientFileName ] );
    set( handles.displayFrontPostAmbientFile, 'Enable', 'on' );
end

% Update handles structure
guidata( hObject, handles );

% --- Executes on selection change in controlFrontPostAmbientChannel.
function controlFrontPostAmbientChannel_Callback(hObject, eventdata, handles)
handles.frontPostAmbientChannel = get( hObject, 'Value' );

% Update handles structure
guidata( hObject, handles );

function pushbuttonSelectTrigger_Callback(hObject, ~, handles)

getFileFilter = '*.wav';

if isfield( handles, 'getFileFilter' )
    getFileFilter = handles.getFileFilter;
end

[ handles.triggerFileName, handles.triggerPathName ] = ...
    uigetfile( getFileFilter, 'Load **Trigger** File' );

if handles.triggerFileName ~= 0
    set( handles.displayTriggerFile, 'String', [ handles.triggerPathName, '\', handles.triggerFileName ] ) ;
    set( handles.displayTriggerFile, 'Enable', 'on' );
end

% Update handles structure
guidata(hObject, handles);

% --- Executes on selection change in controlTriggerChannel.
function controlTriggerChannel_Callback(hObject, ~, handles)
handles.triggerChannel = get( handles.controlTriggerChannel, 'Value' );

```

```
538   % Update handles structure
539   guidata( hObject, handles );
540
541   % --- Executes on button press in pushbuttonImportData.
542   function pushbuttonImportData_Callback(hObject, ~, handles)
543
544   set( handles.displayDataImportMessages, 'String', '');
545
546   % Display error if operating condition not selected
547   if ~ isfield(handles, 'operatingCondition') ;
548       set( handles.displayDataImportMessages, 'String', 'Select Operating Condition', 'ForegroundColor',[0.847,0.161,0]
      );
549   end
550
551   % Get info from GUI
552   if ~ isfield( handles, 'opCond' )
553       contents = cellstr( get( handles.controlOperatingCondition, 'String' ) );
554       handles.opCond = contents{ get( handles.controlOperatingCondition, 'Value' ) };
555   end
556
557   if ~ isfield( handles, 'sampleID' )
558       handles.sampleID = get( handles.editSampleID, 'String' );
559   end
560
561   if ~ isfield( handles, 'driverChannel' )
562       handles.driverChannel = get( handles.controlDriverSignalChannel, 'Value' );
563   end
564
565   if ~ isfield( handles, 'driverPreAmbientChannel' )
566       handles.driverPreAmbientChannel = get( handles.controlDriverPreAmbientChannel, 'Value' );
567   end
568
569   if ~ isfield( handles, 'driverPostAmbientChannel' )
570       handles.driverPostAmbientChannel = get( handles.controlDriverPostAmbientChannel, 'Value' );
571   end
572
573   if ~ isfield( handles, 'passengerChannel' )
574       handles.passengerChannel = get( handles.controlPassengerSignalChannel, 'Value' );
575   end
576
577   if ~ isfield( handles, 'passengerPreAmbientChannel' )
578       handles.passengerPreAmbientChannel = get( handles.controlPassengerPreAmbientChannel, 'Value' );
579   end
580
581   if ~ isfield( handles, 'passengerPostAmbientChannel' )
582       handles.passengerPostAmbientChannel = get( handles.controlPassengerPostAmbientChannel, 'Value' );
583   end
584
585   if ~ isfield( handles, 'frontChannel' )
586       handles.frontChannel = get( handles.controlFrontSignalChannel, 'Value' );
```

```
587   end
588
589   if ~ isfield( handles, 'frontPreAmbientChannel' )
590       handles.frontPreAmbientChannel = get( handles.controlFrontPreAmbientChannel, 'Value' );
591   end
592
593   if ~ isfield( handles, 'frontPostAmbientChannel' )
594       handles.frontPostAmbientChannel = get( handles.controlFrontPostAmbientChannel, 'Value' );
595   end
596
597   if ~ isfield( handles, 'triggerChannel' )
598       handles.triggerChannel = get( handles.controlTriggerChannel, 'Value' );
599   end
600
601   %Set up variables
602   switch handles.opCond
603       case 'Stationary'
604           opCond = nominal( ' Stationary' );
605           table = 'tableStationary';
606       case '10 km/h'
607           opCond = nominal( ' 10 km/h' );
608           table = 'table10kmh';
609       case '20 km/h'
610           opCond = nominal( ' 20 km/h' );
611           table = 'table20kmh';
612       case '30 km/h'
613           opCond = nominal( ' 30 km/h' );
614           table = 'table30kmh';
615       case 'Reverse'
616           opCond = nominal( ' Reverse' );
617           table = 'tableReverse';
618   end
619
620   %Pull in data
621   ink = getappdata( 0, 'ink' );
622
623   if isempty( ink )
624       ink = 1;
625   end
626
627   if strcmp( handles.sampleID, 'NA' ) || isempty ( handles.sampleID ) ; %If user does not input sample ID and no data
      has been entered, use ink to define sample ID
628       if isempty( handles.data ) ;
629           handles.sampleID = num2str( ink );
630       else
631           handles.sampleID = num2str( max ( str2num ( char (handles.data.sampleID ) ) )+ 1 ) ; % If data already
              exists, increment the max sample number
632       end
633   end
634
```

```
635   % Checking to make sure all mics have signal and ambient file selected
636   if isfield( handles, 'driverSignalPathName' ) && isfield( handles, 'driverPreAmbientPathName' ) && isfield( handles,
      'driverPostAmbientPathName' ) && isfield( handles, 'passengerSignalPathName' ) ...
637           && isfield( handles, 'passengerPreAmbientPathName' )  && isfield( handles, 'passengerPostAmbientPathName' )&&
          ~isempty( handles.driverSignalPathName ) && ~isempty( handles.driverPreAmbientPathName' ) && ~isempty( handles
          .driverPostAmbientPathName' ) && ~isempty( handles.passengerSignalPathName ) ...
638           && ~isempty( handles.passengerPreAmbientPathName )&& ~isempty( handles.passengerPostAmbientPathName )
639
640       handles = dataImport( handles ); % import into handles.data
641
642       %If stationary OC, import front data
643       if strcmp( char(opCond), 'Stationary' )
644
645           %Error messages if field not selected properly
646           if ~ isfield( handles, 'frontSignalPathName' ) || isempty ( handles.frontSignalPathName ) ;
647               set( handles.displayDataImportMessages, 'String', 'Select signal file for Front Mic', 'ForegroundColor',[
                  0.847,0.161,0]);
648           elseif handles.frontSignalPathName == 0
649               set( handles.displayDataImportMessages, 'String', 'Select signal file for Front Mic', 'ForegroundColor',[
                  0.847,0.161,0]);
650
651           elseif ~ isfield( handles, 'frontPreAmbientPathName' ) || isempty ( handles.frontPreAmbientPathName ) ;
652               set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Front Mic',
                  'ForegroundColor',[0.847,0.161,0]);
653           elseif handles.frontPreAmbientPathName == 0
654               set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Front Mic',
                  'ForegroundColor',[0.847,0.161,0]);
655
656           elseif ~ isfield( handles, 'frontPostAmbientPathName' ) || isempty ( handles.frontPostAmbientPathName ) ;
657               set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Front Mic',
                  'ForegroundColor',[0.847,0.161,0]);
658           elseif handles.frontPostAmbientPathName == 0
659               set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Front Mic',
                  'ForegroundColor',[0.847,0.161,0]);
660
661           end
662
663           handles = frontImport( handles ); % import into handles.data
664       end
665
666       % if passby, import trigger data
667       if strcmp( char(opCond), '10 km/h' ) || strcmp( char(opCond), '20 km/h' )|| strcmp( char(opCond), '30 km/h' )
668           if ~ isfield( handles, 'triggerFileName' ) || isempty ( handles.triggerFileName ) ;
669               set( handles.displayDataImportMessages, 'String', 'Select trigger file', 'ForegroundColor',[0.847,0.161,0
                  ]);
670           elseif ( handles.triggerFileName ) == 0
671               set( handles.displayDataImportMessages, 'String', 'Select trigger file', 'ForegroundColor',[0.847,0.161,0
                  ]);
672           end
673           handles = triggerImport( handles );  % import into handles.data
```

```
674         end
675
676         handles = displayTable( handles, opCond, table ); % display in tables
677         handles = resetDisplays( handles ); % reset file selections
678
679         tableStationarySampleList = unique( handles.data( handles.data.opCond == 'Stationary' , { 'sampleID' } ) );
680         table10kmhSampleList = unique( handles.data( handles.data.opCond == '10 km/h' , { 'sampleID' } ) );
681         table20kmhSampleList = unique( handles.data( handles.data.opCond == '20 km/h' , { 'sampleID' } ) );
682         table30kmhSampleList = unique( handles.data( handles.data.opCond == '30 km/h' , { 'sampleID' } ) );
683         tableReverseSampleList = unique( handles.data( handles.data.opCond == 'Reverse' , { 'sampleID' } ) );
684
685         if ~handles.useTestData % If not using test data, check to see if there are at least 4 samples for each operating
            condition. Do not enable Submit Data button until 4 samples for each OC
686             if size (tableStationarySampleList, 1 ) >= 4 && size (table10kmhSampleList, 1 ) >= 4 && size (
                table20kmhSampleList, 1 ) >= 4 ...
687                     && size (table30kmhSampleList, 1 ) >= 4 && size (tableReverseSampleList, 1 ) >= 4
688                 set( handles.buttonSubmitData, 'Enable', 'on' );
689                 set( handles.textDataSubmittalMessage, 'String', ' ' );
690             else
691                 set( handles.buttonSubmitData, 'Enable', 'off' );
692                 set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating
                    condition to submit data' );
693             end
694         end
695
696         %Updated handles structure
697         guidata( hObject, handles );
698         setappdata( 0, 'ink', ink + 1 );
699         handles.output = handles.data;
700
701     else
702         % Write error messages if fields don't exist
703         if ~isfield( handles, 'driverSignalPathName' )|| isempty ( handles.driverSignalPathName );
704             set( handles.displayDataImportMessages, 'String', 'Select signal file for Driver Side Mic', 'ForegroundColor'
                ,[0.847,0.161,0]);
705
706         elseif handles.driverSignalPathName == 0
707             set( handles.displayDataImportMessages, 'String', 'Select signal file for Driver Side Mic', 'ForegroundColor'
                ,[0.847,0.161,0]);
708
709         elseif ~isfield( handles, 'driverPreAmbientPathName' )|| isempty ( handles.driverPreAmbientPathName );
710             set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Driver Side Mic',
                'ForegroundColor',[0.847,0.161,0])
711
712         elseif handles.driverPreAmbientPathName == 0
713             set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Driver Side Mic',
                'ForegroundColor',[0.847,0.161,0])
714
715         elseif ~isfield( handles, 'driverPostAmbientPathName' )|| isempty ( handles.driverPostAmbientPathName );
716             set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Driver Side Mic',
```

```
        'ForegroundColor',[0.847,0.161,0])
717
718     elseif handles.driverPostAmbientPathName == 0
719         set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Driver Side Mic',
        'ForegroundColor',[0.847,0.161,0])
720
721     elseif ~isfield( handles, 'passengerSignalPathName' )|| isempty ( handles.passengerSignalPathName );
722         set( handles.displayDataImportMessages, 'String', 'Select signal file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0])
723
724     elseif handles.passengerSignalPathName == 0
725         set( handles.displayDataImportMessages, 'String', 'Select signal file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0])
726
727     elseif ~isfield( handles, 'passengerPreAmbientPathName' )|| isempty ( handles.passengerPreAmbientPathName );
728         set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0]);
729
730     elseif handles.passengerPreAmbientPathName == 0
731         set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0])
732
733     elseif ~isfield( handles, 'passengerPostAmbientPathName' )|| isempty ( handles.passengerPostAmbientPathName );
734         set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0]);
735
736     elseif handles.passengerPostAmbientPathName == 0
737         set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0])
738     end
739 end;
740
741 % --- Executes on button press in pushbuttonClearStationary.
742 function pushbuttonClearStationary_Callback(hObject, ~, handles)
743 % clear data in stationary table and remove from handles.data
744
745 set( handles.tableStationary, 'Data', cell( 4, 3 ) );
746 handles.data( handles.data.opCond == 'Stationary', : ) = [];
747 handles.datStationary = [];
748
749 if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 ) >= 4 ...
750         && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
751     set( handles.buttonSubmitData, 'Enable', 'on' );
752     set( handles.textDataSubmittalMessage, 'String', ' ' );
753
754 else
755     set( handles.buttonSubmitData, 'Enable', 'off' );
756     set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to
    submit data' );
757 end
```

```
758
759   %Update gui handles
760   guidata( hObject, handles )
761
762   % --- Executes on button press in pushbuttonClear10kmh.
763   function pushbuttonClear10kmh_Callback(hObject, ~, handles)
764   % clear data in 10kmh table and remove from handles.data
765
766   set( handles.table10kmh, 'Data', cell( 4, 3 ) );
767   handles.data( handles.data.opCond == '10 km/h', : ) = [];
768   handles.dat10kmh = [];
769
770   if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 ) >= 4 ...
771           && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
772       set( handles.buttonSubmitData, 'Enable', 'on' );
773       set( handles.textDataSubmittalMessage, 'String', ' ' );
774   else
775       set( handles.buttonSubmitData, 'Enable', 'off' );
776       set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to
          submit data' );
777   end
778
779   %Update gui handles
780   guidata( hObject, handles )
781
782   % --- Executes on button press in pushbuttonClear20kmh.
783   function pushbuttonClear20kmh_Callback(hObject, ~, handles)
784   % clear data in 20kmh table and remove from handles.data
785
786   set( handles.table20kmh, 'Data', cell( 4, 3 ) );
787   handles.data( handles.data.opCond == '20 km/h', : ) = [];
788   handles.dat20kmh = [];
789
790   if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 ) >= 4 ...
791           && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
792       set( handles.buttonSubmitData, 'Enable', 'on' );
793       set( handles.textDataSubmittalMessage, 'String', ' ' );
794   else
795       set( handles.buttonSubmitData, 'Enable', 'off' );
796       set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to
          submit data' );
797   end
798
799   %Update gui handles
800   guidata( hObject, handles )
801
802   % --- Executes on button press in pushbuttonClear30kmh.
803   function pushbuttonClear30kmh_Callback(hObject, ~, handles)
804   % clear data in 30kmh table and remove from handles.data
805
```

```
806   set( handles.table30kmh, 'Data', cell( 4, 3 ) );
807   handles.data( handles.data.opCond == '30 km/h', : ) = [];
808   handles.dat30kmh = [];
809
810   if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 ) >= 4 ...
811           && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
812       set( handles.buttonSubmitData, 'Enable', 'on' );
813       set( handles.textDataSubmittalMessage, 'String', ' ' );
814   else
815       set( handles.buttonSubmitData, 'Enable', 'off' );
816       set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to
          submit data' );
817   end
818
819   %Update gui handles
820   guidata( hObject, handles )
821
822   % --- Executes on button press in pushbuttonClearReverse.
823   function pushbuttonClearReverse_Callback(hObject, ~, handles)
824   % clear data in reverse table and remove from handles.data
825
826   set( handles.tableReverse, 'Data', cell( 4, 3 ) );
827   handles.data( handles.data.opCond == 'Reverse', : ) = [];
828
829   %Update gui handles
830   guidata( hObject, handles )
831
832   % --- Executes on button press in buttonSubmitData.
833   function buttonSubmitData_Callback(~, ~, handles)
834   % hObject    handle to buttonSubmitData (see GCBO)
835   % eventdata  reserved - to be defined in a future version of MATLAB
836   % handles    structure with handles and user data (see GUIDATA)
837
838   % ***********FOR TESTING PURPOSES ONLY************
839   if handles.useTestData
840       load('TestData.mat')
841       setappdata( 0, 'data', handles.data )
842
843       handles.loadedTableDataFlag = 1 ;
844       setappdata( 0, 'loadedTableDataFlag', handles.loadedTableDataFlag ) ;
845   % ***********For actual application (not testing)************
846   else
847       % Save current table data for persistence between GUI windows
848       handles.loadedTableDataFlag = 1 ;
849       handles.loadedTableDataStationary = get(handles.tableStationary, 'data') ;
850       handles.loadedTableData10kmh = get(handles.table10kmh, 'data') ;
851       handles.loadedTableData20kmh = get(handles.table20kmh, 'data') ;
852       handles.loadedTableData30kmh = get(handles.table30kmh, 'data') ;
853       handles.loadedTableDataReverse = get(handles.tableReverse, 'data') ;
854
```

```
855         setappdata( 0, 'loadedTableDataFlag', handles.loadedTableDataFlag ) ;
856         setappdata( 0, 'loadedTableDataStationary', handles.loadedTableDataStationary ) ;
857         setappdata( 0, 'loadedTableData10kmh', handles.loadedTableData10kmh ) ;
858         setappdata( 0, 'loadedTableData20kmh', handles.loadedTableData20kmh ) ;
859         setappdata( 0, 'loadedTableData30kmh', handles.loadedTableData30kmh ) ;
860         setappdata( 0, 'loadedTableDataReverse', handles.loadedTableDataReverse ) ;
861
862         setappdata( 0, 'data', handles.data);
863
864         %Reset displays
865         resetDisplays( handles );
866     end
867
868     close
869     uistack( GUI_mainFinalRule_Option1, 'top' );
870
871     % --- Executes on button press in pushbuttonExit.
872     function pushbuttonExit_Callback(~, ~, handles)
873
874     resetDisplays( handles );
875     close
876
877     uistack( GUI_mainFinalRule_Option1, 'top' );
878
879     % --- Outputs from this function are returned to the command line.
880     function varargout = GUI_fileLoad2_OutputFcn(~, ~, handles)
881
882     % Get default command line output from handles structure
883     varargout{1} = handles.output;
884
885     %*********************************************************************
886     %*********************************************************************
887     %      Helper Functions
888     %*********************************************************************
889     %*********************************************************************
890
891     function handles = resetDisplays( handles )
892     % Reset File / Path Names
893     handles.driverSignalPathName = [];
894     handles.driverSignalFileName = [];
895
896     set( handles.displayDriverSignalFile, 'String', 'NA' );
897     set( handles.displayDriverSignalFile, 'Enable', 'off' );
898
899     handles.driverPreAmbientPathName = [];
900     handles.driverPreAmbientFileName = [];
901
902     handles.driverPostAmbientPathName = [];
903     handles.driverPostAmbientFileName = [];
904
```

```
905   set( handles.displayDriverPreAmbientFile, 'String', 'NA' );
906   set( handles.displayDriverPreAmbientFile, 'Enable', 'off' );
907
908   set( handles.displayDriverPostAmbientFile, 'String', 'NA' );
909   set( handles.displayDriverPostAmbientFile, 'Enable', 'off' );
910
911   handles.passengerSignalPathName = [];
912   handles.passengerSignalFileName = [];
913
914   set( handles.displayPassengerSignalFile, 'String', 'NA' );
915   set( handles.displayPassengerSignalFile, 'Enable', 'off' );
916
917   handles.passengerPreAmbientPathName = [];
918   handles.passengerPreAmbientFileName = [];
919
920   handles.passengerPostAmbientPathName = [];
921   handles.passengerPostAmbientFileName = [];
922
923   set( handles.displayPassengerPreAmbientFile, 'String', 'NA' );
924   set( handles.displayPassengerPreAmbientFile, 'Enable', 'off' );
925
926   set( handles.displayPassengerPostAmbientFile, 'String', 'NA' );
927   set( handles.displayPassengerPostAmbientFile, 'Enable', 'off' );
928
929   handles.frontSignalPathName = [];
930   handles.frontSignalFileName = [];
931
932   set( handles.displayFrontSignalFile, 'String', 'NA' );
933   set( handles.displayFrontSignalFile, 'Enable', 'off' );
934
935   handles.frontPreAmbientPathName = [];
936   handles.frontPreAmbientFileName = [];
937
938   handles.frontPostAmbientPathName = [];
939   handles.frontPostAmbientFileName = [];
940
941   set( handles.displayFrontPreAmbientFile, 'String', 'NA' );
942   set( handles.displayFrontPreAmbientFile, 'Enable', 'off' );
943
944   set( handles.displayFrontPostAmbientFile, 'String', 'NA' );
945   set( handles.displayFrontPostAmbientFile, 'Enable', 'off' );
946
947   handles.triggerPathName = [];
948   handles.triggerFileName = [];
949
950   set( handles.displayTriggerFile, 'String', 'NA' );
951   set( handles.displayTriggerFile, 'Enable', 'off' );
952
953   handles.sampleID = [];
954   set( handles.editSampleID, 'String', 'NA' );
```

```
955
956   function handles = dataImport( handles )
957   % Import data into handles.data
958
959   opCond = handles.opCond;
960
961   dataImport = dataset( { { opCond; opCond; opCond; opCond; opCond; opCond }, 'opCond' }, ...
962       { { 'driver'; 'driver'; 'driver'; 'passenger'; 'passenger'; 'passenger' }, 'micSide' }, ...
963       { { handles.sampleID; handles.sampleID; handles.sampleID; handles.sampleID; handles.sampleID; handles.sampleID },
          'sampleID' }, ...
964       { [ handles.driverChannel; handles.driverPreAmbientChannel; handles.driverPostAmbientChannel; handles.
          passengerChannel; handles.passengerPreAmbientChannel; handles.passengerPostAmbientChannel ], 'channel' }, ...
965       { { 'signal'; 'preAmbient'; 'postAmbient'; 'signal'; 'preAmbient'; 'postAmbient' }, 'signalType' }, ...
966       { { handles.driverSignalPathName; handles.driverPreAmbientPathName; handles.driverPostAmbientPathName; handles.
          passengerSignalPathName; handles.passengerPreAmbientPathName; handles.passengerPostAmbientPathName }, 'pathName'
          }, ...
967       { { handles.driverSignalFileName; handles.driverPreAmbientFileName; handles.driverPostAmbientFileName; handles.
          passengerSignalFileName; handles.passengerPreAmbientFileName; handles.passengerPostAmbientFileName }, 'fileName'
          }, ...
968       { { []; []; []; []; []; [] }, 'TriggerThreshold' }, ...
969       { { []; []; []; []; []; [] }, 'idx' } );
970
971   dataImport.opCond = nominal( dataImport.opCond );
972   dataImport.micSide = nominal( dataImport.micSide );
973   dataImport.sampleID = nominal( dataImport.sampleID );
974   dataImport.signalType = nominal( dataImport.signalType );
975   dataImport.pathName = nominal( dataImport.pathName );
976   dataImport.fileName = nominal( dataImport.fileName );
977
978   if isempty( handles.data )
979       handles.data = dataImport;
980   else
981       handles.data = [ handles.data; dataImport ];
982   end
983
984   function handles = frontImport( handles )
985   % import front data into handles.data
986
987   opCond = handles.opCond;
988
989   frontImport = dataset( { { opCond; opCond; opCond }, 'opCond' }, ...
990       { { 'front'; 'front'; 'front' }, 'micSide' }, ...
991       { { handles.sampleID; handles.sampleID; handles.sampleID }, 'sampleID' }, ...
992       { [ handles.frontChannel; handles.frontPreAmbientChannel; handles.frontPostAmbientChannel ], 'channel' }, ...
993       { { 'signal'; 'preAmbient'; 'postAmbient' }, 'signalType' }, ...
994       { { handles.frontSignalPathName; handles.frontPreAmbientPathName; handles.frontPostAmbientPathName }, 'pathName'
          }, ...
995       { { handles.frontSignalFileName; handles.frontPreAmbientFileName; handles.frontPostAmbientFileName }, 'fileName'
          }, ...
996       { { []; []; [] }, 'TriggerThreshold' }, ...
```

```
    { { []; []; [] }, 'idx' } );

frontImport.opCond = nominal( frontImport.opCond );
frontImport.micSide = nominal( frontImport.micSide );
frontImport.sampleID = nominal( frontImport.sampleID );
frontImport.signalType = nominal( frontImport.signalType );
frontImport.pathName = nominal( frontImport.pathName );
frontImport.fileName = nominal( frontImport.fileName );

handles.data = [ handles.data; frontImport ];

function handles = triggerImport( handles )
%import trigger data into handles.data

opCond = handles.opCond;

triggerImport = dataset( { { opCond }, 'opCond' }, ...
    { { 'all' }, 'micSide' }, ...
    { { handles.sampleID }, 'sampleID' }, ...
    { handles.triggerChannel, 'channel' }, ...
    { { 'trigger' }, 'signalType' }, ...
    { { handles.triggerPathName }, 'pathName' }, ...
    { { handles.triggerFileName }, 'fileName' }, ...
    { { [20] }, 'TriggerThreshold' }, ...
    { { [] }, 'idx' } );

triggerImport.opCond = nominal( triggerImport.opCond );
triggerImport.micSide = nominal( triggerImport.micSide );
triggerImport.sampleID = nominal( triggerImport.sampleID );
triggerImport.signalType = nominal( triggerImport.signalType );
triggerImport.pathName = nominal( triggerImport.pathName );
triggerImport.fileName = nominal( triggerImport.fileName );

handles.data = [ handles.data; triggerImport ];

function handles = displayTable( handles, opCond, table )
% Display file information in tables

try
    handles.fileLoadCondition = getappdata( 0, 'fileLoadCondition')  ;
catch
    handles.fileLoadCondition = 'individual' ;
end

if strcmp( handles.fileLoadCondition, 'batch' )
    fullOpCondList = { 'Stationary' , '10 km/h', '20 km/h', '30 km/h' 'Reverse'} ;

    for ink = 1 : length( fullOpCondList )
        opCond = fullOpCondList{ ink };

```

```
1047            if strcmp( cellstr( opCond ), 'Stationary' )
1048                dat = handles.datStationary ;
1049                table = 'tableStationary' ;
1050            end
1051
1052            if strcmp( cellstr( opCond ), '10 km/h' )
1053                dat = handles.dat10kmh ;
1054                table = 'table10kmh' ;
1055            end
1056
1057            if strcmp( cellstr( opCond ), '20 km/h' )
1058                dat = handles.dat20kmh ;
1059                table = 'table20kmh' ;
1060            end
1061
1062            if strcmp( cellstr( opCond ), '30 km/h' )
1063                dat = handles.dat30kmh ;
1064                table = 'table30kmh' ;
1065            end
1066
1067            if strcmp( cellstr( opCond ), 'Reverse' )
1068                dat = handles.datReverse ;
1069                table = 'tableReverse' ;
1070            end
1071
1072            set( handles.( table ), 'Data', dat );
1073        end
1074        % Enable the submit data button only if 4 samples exist for each operating condition
1075            if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 )
                >= 4 ...
1076                    && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
1077                set( handles.buttonSubmitData, 'Enable', 'on' );
1078                set( handles.textDataSubmittalMessage, 'String', ' ' );
1079
1080            else
1081                set( handles.buttonSubmitData, 'Enable', 'off' );
1082                set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating
                    condition to submit data' );
1083            end
1084    else
1085        sampleList = unique( handles.data( handles.data.opCond == opCond , { 'sampleID' } ) );
1086        sampleList = sort( sampleList.sampleID );
1087
1088        if ~isempty( sampleList )
1089
1090            for chalk = 1: length( sampleList )
1091
1092                if isempty( handles.data( handles.data.opCond == opCond & ...
1093                        handles.data.sampleID == sampleList( chalk ) & ...
1094                        handles.data.signalType == 'trigger', { 'fileName' } ) );
```

```
1095            else
1096                trigger = 'true';
1097                triggerSignal = handles.data( handles.data.opCond == opCond & ...
1098                    handles.data.sampleID == sampleList( chalk ) & ...
1099                    handles.data.signalType == 'trigger', { 'fileName' } ) ;
1100            end
1101
1102            driverSignal = handles.data( handles.data.opCond == opCond & ...
1103                handles.data.sampleID == sampleList( chalk ) &  ...
1104                handles.data.micSide == 'driver' & ...
1105                handles.data.signalType == 'signal', { 'fileName' } );
1106            driverPreAmbient = handles.data( handles.data.opCond == opCond & ...
1107                handles.data.sampleID == sampleList( chalk ) &  ...
1108                handles.data.micSide == 'driver' & ...
1109                handles.data.signalType == 'preAmbient', { 'fileName' } );
1110            driverPostAmbient = handles.data( handles.data.opCond == opCond & ...
1111                handles.data.sampleID == sampleList( chalk ) &  ...
1112                handles.data.micSide == 'driver' & ...
1113                handles.data.signalType == 'postAmbient', { 'fileName' } );
1114            passengerSignal = handles.data( handles.data.opCond == opCond & ...
1115                handles.data.sampleID == sampleList( chalk ) &  ...
1116                handles.data.micSide == 'passenger' & ...
1117                handles.data.signalType == 'signal', { 'fileName' } );
1118            passengerPreAmbient = handles.data( handles.data.opCond == opCond & ...
1119                handles.data.sampleID == sampleList( chalk ) &  ...
1120                handles.data.micSide == 'passenger' & ...
1121                handles.data.signalType == 'preAmbient', { 'fileName' } );
1122            passengerPostAmbient = handles.data( handles.data.opCond == opCond & ...
1123                handles.data.sampleID == sampleList( chalk ) &  ...
1124                handles.data.micSide == 'passenger' & ...
1125                handles.data.signalType == 'postAmbient', { 'fileName' } );
1126            frontSignal = handles.data( handles.data.opCond == opCond & ...
1127                handles.data.sampleID == sampleList( chalk ) &  ...
1128                handles.data.micSide == 'front' & ...
1129                handles.data.signalType == 'signal', { 'fileName' } );
1130            frontPreAmbient = handles.data( handles.data.opCond == opCond & ...
1131                handles.data.sampleID == sampleList( chalk ) &  ...
1132                handles.data.micSide == 'front' & ...
1133                handles.data.signalType == 'preAmbient', { 'fileName' } );
1134            frontPostAmbient = handles.data( handles.data.opCond == opCond & ...
1135                handles.data.sampleID == sampleList( chalk ) &  ...
1136                handles.data.micSide == 'front' & ...
1137                handles.data.signalType == 'postAmbient', { 'fileName' } );
1138
1139            if strcmp( cellstr( opCond ), 'Stationary' )
1140                dat( chalk, : ) = [ cellstr( sampleList( chalk ) ),...
1141                    cellstr( driverSignal.fileName ), cellstr( driverPreAmbient.fileName ), cellstr( driverPostAmbient
                    .fileName ),...
1142                    cellstr( passengerSignal.fileName ), cellstr( passengerPreAmbient.fileName ), cellstr(
                    passengerPostAmbient.fileName ),...
```

```
                    cellstr( frontSignal.fileName ), cellstr( frontPreAmbient.fileName ), cellstr( frontPostAmbient.
                    fileName )];

            elseif strcmp( cellstr( opCond ), 'Reverse' )
                dat( chalk, : ) = [ cellstr( sampleList( chalk ) ),...
                    cellstr( driverSignal.fileName ), cellstr( driverPreAmbient.fileName ), cellstr( driverPostAmbient
                    .fileName ),...
                    cellstr( passengerSignal.fileName ), cellstr( passengerPreAmbient.fileName ), cellstr(
                    passengerPostAmbient.fileName )];
            else
                dat( chalk, : ) = [ cellstr( sampleList( chalk ) ),...
                    cellstr( driverSignal.fileName ), cellstr( driverPreAmbient.fileName ), cellstr( driverPostAmbient
                    .fileName ),...
                    cellstr( passengerSignal.fileName ), cellstr( passengerPreAmbient.fileName ), cellstr(
                    passengerPostAmbient.fileName ),...
                    cellstr( triggerSignal.fileName ) ];
            end
        end
        set( handles.( table ), 'Data', dat );
    end
end


%**************************************************************************
%**************************************************************************
%      Create Functions
%**************************************************************************
%**************************************************************************

% --- Executes during object creation, after setting all properties.
function buttonSubmitData_CreateFcn(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function controlOperatingCondition_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayDriverSignalFile_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlDriverSignalChannel_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```
% --- Executes during object creation, after setting all properties.
function displayDriverPreAmbientFile_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlDriverPreAmbientChannel_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayPassengerSignalFile_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlPassengerSignalChannel_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayPassengerPreAmbientFile_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlPassengerPreAmbientChannel_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayFrontSignalFile_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayFrontPreAmbientFile_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlFrontSignalChannel_CreateFcn(hObject, ~, ~)
```

```
1238  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1239      set(hObject,'BackgroundColor','white');
1240  end
1241
1242  % --- Executes during object creation, after setting all properties.
1243  function controlFrontPreAmbientChannel_CreateFcn(hObject, ~, ~)
1244  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1245      set(hObject,'BackgroundColor','white');
1246  end
1247
1248  % --- Executes during object creation, after setting all properties.
1249  function controlTriggerChannel_CreateFcn(hObject, ~, ~)
1250  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1251      set(hObject,'BackgroundColor','white');
1252  end
1253
1254  % --- Executes during object creation, after setting all properties.
1255  function displayTriggerFile_CreateFcn(hObject, ~, ~)
1256  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1257      set(hObject,'BackgroundColor','white');
1258  end
1259
1260  % --- Executes during object creation, after setting all properties.
1261  function controlDriverPostAmbientChannel_CreateFcn(hObject, eventdata, handles)
1262  % hObject    handle to controlDriverPostAmbientChannel (see GCBO)
1263  % eventdata  reserved - to be defined in a future version of MATLAB
1264  % handles    empty - handles not created until after all CreateFcns called
1265
1266  % Hint: popupmenu controls usually have a white background on Windows.
1267  %       See ISPC and COMPUTER.
1268  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1269      set(hObject,'BackgroundColor','white');
1270  end
1271
1272  % --- Executes during object creation, after setting all properties.
1273  function displayDriverPostAmbientFile_CreateFcn(hObject, eventdata, handles)
1274  % hObject    handle to displayDriverPostAmbientFile (see GCBO)
1275  % eventdata  reserved - to be defined in a future version of MATLAB
1276  % handles    empty - handles not created until after all CreateFcns called
1277
1278  % Hint: edit controls usually have a white background on Windows.
1279  %       See ISPC and COMPUTER.
1280  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1281      set(hObject,'BackgroundColor','white');
1282  end
1283
1284  % --- Executes during object creation, after setting all properties.
1285  function OperatingConditionFileLoadMenu_CreateFcn(hObject, eventdata, handles)
1286  % hObject    handle to OperatingConditionFileLoadMenu (see GCBO)
1287  % eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlPassengerPostAmbientChannel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to controlPassengerPostAmbientChannel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayPassengerPostAmbientFile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to displayPassengerPostAmbientFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlFrontPostAmbientChannel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to controlFrontPostAmbientChannel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayFrontPostAmbientFile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to displayFrontPostAmbientFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

```

```
1338   % Hint: edit controls usually have a white background on Windows.
1339   %       See ISPC and COMPUTER.
1340   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1341       set(hObject,'BackgroundColor','white');
1342   end
1343
1344   %**************************************************************************
1345   %**************************************************************************
1346   %      Empty Callbacks
1347   %**************************************************************************
1348   %**************************************************************************
1349   function displayPassengerPreAmbientFile_Callback(hObject, eventdata, handles)
1350   function displayDriverPreAmbientFile_Callback(hObject, eventdata, handles)
1351   function displayDriverPostAmbientFile_Callback(hObject, eventdata, handles)
1352   function displayFrontPreAmbientFile_Callback(hObject, eventdata, handles)
1353   function displayPassengerPostAmbientFile_Callback(hObject, eventdata, handles)
1354   function displayFrontPostAmbientFile_Callback(hObject, eventdata, handles)
1355
```

```
%This is the main entry point for the NHTSA QUIET VEHICLE COMPLIANCE TOOL

function varargout = GUI_mainFinalRule_Option1(varargin)
% This application will allow the user to import wav files with passby,
% trigger and stationary data. The user then enters information about
% channels, singal type and trial number. The user can preview the signal
% in order to examine its quality or to identify extraneuous sounds. Saved
% files can be validated and the SPL can be computed for each file.


%**************************************************************************
%**************************************************************************
%       Creation
%**************************************************************************
%**************************************************************************

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @GUI_mainFinalRule_Option1_OpeningFcn, ...
    'gui_OutputFcn',  @GUI_mainFinalRule_Option1_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before GUI_mainFinalRule_Option1 is made visible.
function GUI_mainFinalRule_Option1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_mainFinalRule_Option1 (see VARARGIN)

% Clear the console display
clc

% Set the name of the GUI window and assign version number (used in reporting)
handles.versionNumber = 'Beta 1';
toolName = sprintf( 'NHTSA Quiet Vehicle Compliance Tool - Version: %s', handles.versionNumber ) ;
set(handles.figure1, 'Name', toolName);
```

```
51
52  % Choose default command line output
53  handles.output = hObject;
54
55  % Initialize some application data
56  handles.pref2 = ( 20e-6 )^2;
57
58  plot( handles.axesDisplay, 1, 1 )
59  axes( handles.axesDisplay ) % Need to explicitly set this or it can change
60  ylabel( 'Acoustic Pressure (Pa)' )
61  set( handles.axesDisplay, 'XScale', 'linear' )
62  axis( [ 0 20 -inf inf ] )
63
64  handles.data = getappdata( 0, 'data' );
65  setappdata( 0, 'ink', 1 );
66  set( handles.controlOperatingCondition, 'Value', 1 );
67
68  if ~ isempty( handles.data );
69      set( handles.displayDriverSignalFile,            'Enable', 'on' );
70      set( handles.displayDriverPreAmbientFile,        'Enable', 'on' );
71      set( handles.displayDriverPostAmbientFile,       'Enable', 'on' );
72      set( handles.displayTriggerFile,                 'Enable', 'on' );
73      set( handles.displayPassengerSignalFile,         'Enable', 'on' );
74      set( handles.displayPassengerPreAmbientFile,     'Enable', 'on' );
75      set( handles.displayPassengerPostAmbientFile,    'Enable', 'on' );
76      set( handles.controlDriverSignalChannel,         'Enable', 'on' );
77      set( handles.controlDriverPreAmbientChannel,     'Enable', 'on' );
78      set( handles.controlDriverPostAmbientChannel,    'Enable', 'on' );
79      set( handles.controlTriggerChannel,              'Enable', 'on' );
80      set( handles.controlPassengerSignalChannel,      'Enable', 'on' );
81      set( handles.controlPassengerPreAmbientChannel,  'Enable', 'on' );
82      set( handles.controlPassengerPostAmbientChannel, 'Enable', 'on' );
83      set( handles.checkboxTrigger,                    'Enable', 'on' );
84      set( handles.pushbuttonDiscardSample,            'Enable', 'on' );
85      set( handles.pushbuttonCompute,                  'Enable', 'on' );
86      set( handles.pushbuttonStop,                     'Enable', 'off' );
87      set( handles.pushbuttonPrevious,                 'Enable', 'on' );
88      set( handles.pushbuttonNext,                     'Enable', 'on' );
89      set( handles.editSampleID,                       'Enable', 'on' );
90      set( handles.editTriggerThreshold,               'Enable', 'on' );
91
92      handles.useTrigger = get( handles.checkboxTrigger, 'Value');
93      if ~ handles.useTrigger
94          set( handles.controlTriggerChannel, 'Enable', 'off' )
95      else
96          set( handles.controlTriggerChannel, 'Enable', 'on' )
97      end
98
99  end
100
```

```
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = GUI_mainFinalRule_Option1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%**************************************************************************
%**************************************************************************
%       Callbacks and Button Presses
%**************************************************************************
%**************************************************************************

% --- Executes on button press in controlFileSelect.
function controlFileSelect_Callback(hObject, eventdata, handles)
% hObject    handle to controlFileSelect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% For batch folder load error
set( handles.textBatchFolderError, 'Visible', 'off' );
handles.folderStructureError = 0 ;
handles.batchFileCancel = false ;

% Determine if individual or batch file load and prepare variables for importing through GUI_fileLoad2
if get( handles.radioFileLoad, 'Value' ) == 1
    handles.fileLoadCondition = 'individual' ;
    setappdata( 0, 'fileLoadCondition', handles.fileLoadCondition )
    handles = GUI_fileLoad2( handles ) ;
else
    handles.fileLoadCondition = 'batch' ;
    handles = selectBatchDirectory( handles ) ;
    if handles.batchFileCancel
        return
    end
    if handles.folderStructureError
        set( handles.textBatchFolderError, 'Visible', 'on' );
        return
    end

    handles = getChannelMapBatchLoad( handles ) ; % Variable assignments based on channel mapping version for batch
    load
    handles = getFileNameFormat( handles ) ; % Creates variables based on file name
```

```
150         handles = dataImportBatch( handles ) ; % Imports metadata, not time signals, has a place holder for the time
            signals
151
152         setappdata( 0, 'fileLoadCondition', handles.fileLoadCondition )
153         setappdata( 0, 'datStationary', handles.datStationary )
154         setappdata( 0, 'dat10kmh', handles.dat10kmh )
155         setappdata( 0, 'dat20kmh', handles.dat20kmh )
156         setappdata( 0, 'dat30kmh', handles.dat30kmh )
157         setappdata( 0, 'datReverse', handles.datReverse )
158
159         handles = GUI_fileLoad2( handles );
160
161     end
162
163     % --- Executes on selection change in controlOperatingCondition.
164     function controlOperatingCondition_Callback(hObject, eventdata, handles)
165     % hObject    handle to controlOperatingCondition (see GCBO)
166     % eventdata  reserved - to be defined in a future version of MATLAB
167     % handles    structure with handles and user data (see GUIDATA)
168
169     % Hints: contents = cellstr(get(hObject,'String')) returns controlOperatingCondition contents as cell array
170     %        contents{get(hObject,'Value')} returns selected item from controlOperatingCondition
171
172     contents = cellstr( get( hObject, 'String' ) ); % Need to get contents, before you can index into them
173     handles.operatingCondition = contents{ get( hObject, 'Value' ) };
174
175     switch handles.operatingCondition
176         case ' Stationary'
177             handles.opCond = 'Stationary';
178         case ' 10 km/h'
179             handles.opCond = '10 km/h';
180         case ' 20 km/h'
181             handles.opCond = '20 km/h';
182         case ' 30 km/h'
183             handles.opCond = '30 km/h';
184         case ' Reverse'
185             handles.opCond = 'Reverse';
186     end
187
188     handles.data = getappdata( 0, 'data' );
189     handles.samplesDone = getappdata( 0, 'samplesDone' );
190
191     handles.currentSample = 1;
192
193     if ~ isempty( handles.data )
194
195         if ~ isempty( handles.data( handles.data.opCond == handles.opCond, : ) )
196
197             %clear all displays
198             set( handles.displayDriverSignalFile,         'String', '') ;
```

```
199         set( handles.displayDriverPreAmbientFile,     'String', '') ;
200         set( handles.displayDriverPostAmbientFile,    'String', '') ;
201         set( handles.displayPassengerSignalFile,      'String', '') ;
202         set( handles.displayPassengerPreAmbientFile,  'String', '') ;
203         set( handles.displayPassengerPostAmbientFile, 'String', '') ;
204         set( handles.displayTriggerFile,              'String', '') ;
205         set( handles.editTriggerThreshold,            'String', '') ;
206         set( handles.displayFrontMicSignalFile,       'String', '') ;
207         set( handles.displayFrontMicPreAmbientFile,   'String', '') ;
208         set( handles.displayFrontMicPostAmbientFile,  'String', '') ;
209
210         %Enable displays according to OC
211         if strcmpi( handles.opCond, 'Stationary' );
212             set( handles.textPP,                'Enable', 'off' );
213             set( handles.textTrSig,             'Enable', 'off' );
214             set( handles.checkboxTrigger,       'Value', 0 );
215             set( handles.checkboxTrigger,       'Enable', 'off' );
216             set( handles.displayTriggerFile,    'Enable', 'off' );
217             set( handles.controlTriggerChannel, 'Enable', 'off' );
218             set( handles.textTriggerTime,       'Enable', 'off' );
219             set( handles.textTrThres,           'Enable', 'off' );
220             set( handles.textTrSec,             'Enable', 'off' );
221             set( handles.editTriggerThreshold,  'Enable', 'off' );
222             set( handles.textTrPercent,         'Enable', 'off' );
223
224             set( handles.textFront,                         'Enable', 'on' );
225             set( handles.textFrSig,                         'Enable', 'on' );
226             set( handles.displayFrontMicSignalFile,         'Enable', 'on' );
227             set( handles.controlFrontMicSignalChannel,      'Enable', 'on' );
228             set( handles.textFrPreAmb,                      'Enable', 'on' );
229             set( handles.textFrPostAmb,                     'Enable', 'on' );
230             set( handles.displayFrontMicPreAmbientFile,     'Enable', 'on' );
231             set( handles.displayFrontMicPostAmbientFile,    'Enable', 'on' );
232             set( handles.controlFrontMicPreAmbientChannel,  'Enable', 'on' );
233             set( handles.controlFrontMicPostAmbientChannel, 'Enable', 'on' );
234
235         elseif  strcmpi( handles.opCond, '10 km/h' );
236             set( handles.textPP,                'Enable', 'on' );
237             set( handles.textTrSig,             'Enable', 'on' );
238             set( handles.checkboxTrigger,       'Value', 1 );
239             set( handles.checkboxTrigger,       'Enable', 'on' );
240             set( handles.displayTriggerFile,    'Enable', 'on' );
241             set( handles.controlTriggerChannel, 'Enable', 'on' );
242             set( handles.textTriggerTime,       'Enable', 'on' );
243             set( handles.textTrThres,           'Enable', 'on' );
244             set( handles.textTrSec,             'Enable', 'on' );
245             set( handles.editTriggerThreshold,  'Enable', 'on' );
246             set( handles.textTrPercent,         'Enable', 'on' );
247
248             set( handles.textFront,                         'Enable', 'off' );
```

```
            set( handles.textFrSig,                          'Enable', 'off' );
            set( handles.displayFrontMicSignalFile,          'Enable', 'off' );
            set( handles.controlFrontMicSignalChannel,       'Enable', 'off' );
            set( handles.textFrPreAmb,                       'Enable', 'off' );
            set( handles.textFrPostAmb,                      'Enable', 'off' );
            set( handles.displayFrontMicPreAmbientFile,      'Enable', 'off' );
            set( handles.controlFrontMicPreAmbientChannel,   'Enable', 'off' );
            set( handles.displayFrontMicPostAmbientFile,     'Enable', 'off' );
            set( handles.controlFrontMicPostAmbientChannel,  'Enable', 'off' );

        elseif  strcmpi( handles.opCond, '20 km/h' );
            set( handles.textPP,                 'Enable', 'on' );
            set( handles.textTrSig,              'Enable', 'on' );
            set( handles.checkboxTrigger,        'Value', 1 );
            set( handles.checkboxTrigger,        'Enable', 'on' );
            set( handles.displayTriggerFile,     'Enable', 'on' );
            set( handles.controlTriggerChannel,  'Enable', 'on' );
            set( handles.textTriggerTime,        'Enable', 'on' );
            set( handles.textTrThres,            'Enable', 'on' );
            set( handles.textTrSec,              'Enable', 'on' );
            set( handles.editTriggerThreshold,   'Enable', 'on' );
            set( handles.textTrPercent,          'Enable', 'on' );

            set( handles.textFront,                          'Enable', 'off' );
            set( handles.textFrSig,                          'Enable', 'off' );
            set( handles.displayFrontMicSignalFile,          'Enable', 'off' );
            set( handles.controlFrontMicSignalChannel,       'Enable', 'off' );
            set( handles.textFrPreAmb,                       'Enable', 'off' );
            set( handles.displayFrontMicPreAmbientFile,      'Enable', 'off' );
            set( handles.controlFrontMicPreAmbientChannel,   'Enable', 'off' );

        elseif  strcmpi( handles.opCond, '30 km/h' );
            set( handles.textPP,                 'Enable', 'on' );
            set( handles.textTrSig,              'Enable', 'on' );
            set( handles.checkboxTrigger,        'Value', 1 );
            set( handles.checkboxTrigger,        'Enable', 'on' );
            set( handles.displayTriggerFile,     'Enable', 'on' );
            set( handles.controlTriggerChannel,  'Enable', 'on' );
            set( handles.textTriggerTime,        'Enable', 'on' );
            set( handles.textTrThres,            'Enable', 'on' );
            set( handles.textTrSec,              'Enable', 'on' );
            set( handles.editTriggerThreshold,   'Enable', 'on' );
            set( handles.textTrPercent,          'Enable', 'on' );

            set( handles.textFront,                          'Enable', 'off' );
            set( handles.textFrSig,                          'Enable', 'off' );
            set( handles.displayFrontMicSignalFile,          'Enable', 'off' );
            set( handles.controlFrontMicSignalChannel,       'Enable', 'off' );
            set( handles.textFrPreAmb,                       'Enable', 'off' );
            set( handles.textFrPostAmb,                      'Enable', 'off' );
```

```
            set( handles.displayFrontMicPreAmbientFile,     'Enable', 'off' );
            set( handles.controlFrontMicPreAmbientChannel,  'Enable', 'off' );
            set( handles.displayFrontMicPostAmbientFile,    'Enable', 'off' );
            set( handles.controlFrontMicPostAmbientChannel, 'Enable', 'off' );

        elseif  strcmpi( handles.opCond, 'Reverse' );
            set( handles.textPP,                'Enable', 'off' );
            set( handles.textTrSig,             'Enable', 'off' );
            set( handles.checkboxTrigger,       'Value', 0 );
            set( handles.checkboxTrigger,       'Enable', 'off' );
            set( handles.displayTriggerFile,    'Enable', 'off' );
            set( handles.controlTriggerChannel, 'Enable', 'off' );
            set( handles.textTriggerTime,       'Enable', 'off' );
            set( handles.textTrThres,           'Enable', 'off' );
            set( handles.textTrSec,             'Enable', 'off' );
            set( handles.editTriggerThreshold,  'Enable', 'off' );
            set( handles.textTrPercent,         'Enable', 'off' );

            set( handles.textFront,                         'Enable', 'off' );
            set( handles.textFrSig,                         'Enable', 'off' );
            set( handles.displayFrontMicSignalFile,         'Enable', 'off' );
            set( handles.controlFrontMicSignalChannel,      'Enable', 'off' );
            set( handles.textFrPreAmb,                      'Enable', 'off' );
            set( handles.textFrPostAmb,                     'Enable', 'off' );
            set( handles.displayFrontMicPreAmbientFile,     'Enable', 'off' );
            set( handles.controlFrontMicPreAmbientChannel,  'Enable', 'off' );
            set( handles.displayFrontMicPostAmbientFile,    'Enable', 'off' );
            set( handles.controlFrontMicPostAmbientChannel, 'Enable', 'off' );
        end

        % Set up sample hierarchy for samples with that operating condition
        numSamples = unique( handles.data( handles.data.opCond == handles.opCond, { 'sampleID' } ) );
        numSamples = sort( numSamples.sampleID );
        handles.numSamples = numSamples ;

        % Display current sample number of total sample numbers
        set( handles.textCurrentSampleNumber, 'String', handles.currentSample );

        % Display number original number of samples
        if ~ isfield( handles, 'discardRows' ) || ~ isempty( handles.discardRows )
            % Display original number of samples (before any discards)
            numSamplesLength = length( numSamples );
            handles.numSamplesLength = numSamplesLength ;
            set( handles.textTotalSampleNumbers, 'String', handles.numSamplesLength  );
            set( handles.textTotalSamplesafterDiscard, 'String', handles.numSamplesLength  );
        end

        % Set the sampleID based on the current sample number
        handles.sampleID = numSamples( handles.currentSample );

```

```
        handles = loadData( handles ); % For Data Review
        handles = plotData( handles ); % For Data Review
    end
end

% Update handles structure
guidata( hObject, handles );

function editTriggerThreshold_Callback(hObject, eventdata, handles)
% hObject    handle to editTriggerThreshold (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editTriggerThreshold as text
%        str2double(get(hObject,'String')) returns contents of editTriggerThreshold as a double

%plot trigger threshold line
ax = handles.ax ;

try
    children = get(ax(2), 'children');
    delete( children(1) );
catch ME
    disp('tigger line does not exist')
end

handles.triggerThresholdValue = str2num( get( handles.editTriggerThreshold, 'String' ) );
handles.trigX1 = min( handles.triggerSignal_Time );
handles.trigX2 = max( handles.triggerSignal_Time );

hold on
handles.hPlotTriggerPercent = plot( ax( 2 ), [ handles.trigX1 handles.trigX2 ], [handles.triggerThresholdValue handles
.triggerThresholdValue], 'color', [0 0.5 0], 'LineStyle', '- -', 'LineWidth', 3) ;

handles.idxAtTriggerThreshold = find( (handles.triggerScaledTo100) > handles.triggerThresholdValue, 1 );
handles.timeAtTriggerThreshold = handles.triggerSignal_Time( handles.idxAtTriggerThreshold );

% Update trigger time display
set( handles.editTrigger, 'String', handles.timeAtTriggerThreshold )

for ink = 1: size ( handles.data, 1 )

    if  strcmp( char(handles.data{ ink, 'signalType' }), 'trigger' ) && handles.data.sampleID(ink) == handles.sampleID
     ;
        handles.data( ink, 'SPL' ) = replacedata( handles.data( ink, 'SPL' ), {handles.triggerThresholdValue} );
    end
end

setappdata( 0, 'data', handles.data);

```

```
% Update handles structure
guidata( hObject, handles );

% --- Executes on button press in pushbuttonPrevious.
function pushbuttonPrevious_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonPrevious (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.currentSample > 1
    % Look into handles.data for desired sample ID, if sample data has been
    % discarded, loop and inc again otherwise, increment by 1

    count = 1 ;

    while handles.currentSample > 1
        temp = handles.data( handles.data.opCond == handles.opCond & handles.data.sampleID == handles.numSamples(
        handles.currentSample - count) , :) ;

        if ~isempty( temp )
            handles.currentSample = handles.currentSample - count ;
            break
        else
            count = count + 1 ;
        end
    end
end

% Update handles structure
guidata( hObject, handles );

plotNextPreviousSamples( handles ) ;

% --- Executes on button press in pushbuttonDiscardSample.
function pushbuttonDiscardSample_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonDiscardSample (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Create the condition for row removal
condition = handles.data.opCond == handles.opCond & handles.data.sampleID == handles.numSamples(handles.currentSample)
 ;
handles.discardRows = handles.data(condition, :) ;

% Remove rows
handles.data (condition,:) = [];

% Update total sample number for analysis after discard
% Display number of samples and total number of samples after
handles.updateNumSamples = unique( handles.data( handles.data.opCond == handles.opCond, { 'sampleID' } ) );
```

```
handles.updateNumSamplesLength = length( handles.updateNumSamples ) ;
set( handles.textTotalSamplesafterDiscard, 'String', handles.updateNumSamplesLength  );

% Advance to next sample
if ( handles.currentSample ) < length( handles.numSamples )
    % While less than max # samples -1, look into handles.data for desired sample ID, if sample data has been
    % discarded, loop and inc again otherwise, increment by 1

    while handles.currentSample < length( handles.numSamples )
        temp = handles.data( handles.data.opCond == handles.opCond & handles.data.sampleID == handles.numSamples(
        handles.currentSample + 1) , :) ;

        if ~isempty( temp )
            handles.currentSample = handles.currentSample + 1 ;
            break
        end
    end
end

% Update handles structure
guidata( hObject, handles );

plotNextPreviousSamples( handles ) ;

% --- Executes on button press in pushbuttonNext.
function pushbuttonNext_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonNext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.currentSample < length( handles.numSamples )
    % while less than max # samples - 1
    % look into handles.data for desired sample ID, if sample data has been
    % discarded, loop and inc again otherwise, increment by 1

    count = 1 ;
    while handles.currentSample < length( handles.numSamples )
        temp = handles.data( handles.data.opCond == handles.opCond & handles.data.sampleID == handles.numSamples(
        handles.currentSample + count) , :) ;

        if ~isempty( temp )
            handles.currentSample = handles.currentSample + count ;
            break
        else
            count = count + 1;
        end
    end
end

% Update handles structure
```

```
493   guidata( hObject, handles );
494
495   plotNextPreviousSamples( handles ) ;
496
497   % --- Executes on button press in pushbuttonCompute.
498   function pushbuttonCompute_Callback(hObject, eventdata, handles)
499   % hObject    handle to pushbuttonCompute (see GCBO)
500   % eventdata  reserved - to be defined in a future version of MATLAB
501   % handles    structure with handles and user data (see GUIDATA)
502
503   % Display appropriate fields and remove results from previous run
504   handles.dataBeforeCompute = handles;
505
506   set( handles.display2dBError,          'Visible', 'off' );
507   set( handles.displayAnalysisComplete,  'Visible', 'off' );
508   set( handles.textAnalysisStopped,      'Visible', 'off' );
509   set( handles.displayCompliance,        'String', 'NA' );
510   set( handles.displayRunDurationValue,  'String', '-' );
511   set( handles.displayCurrentOC,         'Visible', 'on' );
512   set( handles.textRunStatusFor,         'Visible', 'on' );
513   set( handles.displayOutputFilePath,    'String', 'NA'  );
514
515   handles.stopButtonFlag = false ;
516   handles.outputFileCancel = false ;
517
518   datClear = [ '', '', '', '', '' ; ...
519       '', '', '', '', ''; ...
520       '', '', '', '', ''; ...
521       '', '', '', '', ''; ...
522       '', '', '', '', ''; ...
523       '', '', '', '', '' ] ;
524
525   set( handles.uitableResults, 'Data', datClear );
526
527   % Create output directory
528   defaultOutputDirectory = 'C:\NHTSA\QVCT' ;
529
530   if ~exist( defaultOutputDirectory, 'dir' )
531       mkdir( defaultOutputDirectory );
532   end
533
534   % Create results folder based on analysis date
535   analysisDate = date ;
536
537   currentRunOutputFolder = sprintf( '%s/Results_%s', defaultOutputDirectory, analysisDate ) ;
538
539   if ~exist( currentRunOutputFolder, 'dir' )
540       mkdir( currentRunOutputFolder );
541   end
542
```

```
% Prompt user for output directory location
outputDirectory = uigetdir( currentRunOutputFolder, 'Select Folder to Save Results' ) ;

handles.outputDirectory = outputDirectory ;

% Display output directory
set( handles.displayOutputFilePath, 'String', handles.outputDirectory  );

% Compute Analysis
handles.runTic = tic ;
set( handles.pushbuttonCompute, 'Enable', 'off'  );
set( handles.pushbuttonStop,    'Enable', 'on'  );
handles.data = getappdata( 0, 'data' );
handles = main_ComplianceInputPrep( handles );


if handles.stopButtonFlag
    % For stop button
    drawnow
    handles = handles.dataBeforeCompute;
    set( handles.pushbuttonCompute,   'Enable', 'on' );
    set( handles.textAnalysisStopped, 'Visible', 'on' );
    set( handles.pushbuttonStop,      'userdata', 0 )
    set(handles.pushbuttonStop,       'string', 'Stop Analysis')
    return
end

if handles.outputFileCancel
    set( handles.pushbuttonCompute, 'Enable', 'on'  );
    return
end

set( handles.pushbuttonCompute,           'Enable', 'on'  );
set( handles.displayAnalysisComplete,     'Visible', 'on' );
set( handles.displayAnalysisComplete,     'String', 'Analysis Complete' );
set( handles.displayRunStatusCurrentFile, 'Visible', 'off' );
set( handles.textRunStatusOf,             'Visible', 'off' );
set( handles.displayRunStatusTotalFiles,  'Visible', 'off' );

totalDuration = round( toc( handles.runTic ) );
set( handles.displayRunDurationValue, 'String', num2str( totalDuration ) );

set( handles.uitableResults,    'Data', handles.datStatus );
set( handles.displayCompliance, 'String', handles.complianceOverall );
set( handles.pushbuttonStop,    'Enable', 'off'  );

% --- Executes on button press in pushbuttonStop.
function pushbuttonStop_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonStop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
set( handles.pushbuttonStop,'userdata',1)
set( handles.pushbuttonCompute, 'Enable', 'on'  );
set( handles.pushbuttonStop, 'Enable', 'off'  );

%**************************************************************************
%**************************************************************************
%      Helper Functions
%**************************************************************************
%**************************************************************************

function handles = clearFile( handles )

% Reset File / Path Names
if isfield( handles, 'signalPathName' )
    handles.signalPathName = ' NA';
end

if isfield( handles, 'signalFileName' )
    handles.signalFileName = ' NA';
end

if isfield( handles, 'ambientPathName' )
    handles.ambientPathName = ' No Ambient Path';
end

if isfield( handles, 'ambientFileName' )
    handles.ambientFileName = ' No Ambient File';
end

% Reset Displays
if isfield( handles, 'displaySignalPath' )
    set( handles.displaySignalPath, 'String', handles.signalPathName );
end

if isfield( handles, 'displaySignalFile' )
    set( handles.displayDriverSignalFile, 'String', handles.signalFileName );
end

if isfield( handles, 'displayAmbientPath' )
    set( handles.displayAmbientPath, 'String', handles.ambientPathName );
end

if isfield( handles, 'displayAmbientFile' )
    set( handles.displayTriggerFile, 'String', handles.ambientFileName );
end

set( handles.pushbuttonTrim,                    'Enable', 'off' )
set( handles.pushbuttonDiscardSample,           'Enable', 'off' )
set( handles.displaySPLdBA,                     'String', sprintf( '%s', ' NA' ) )
```

```
643   set( handles.controlPlayDriverSignal,            'Enable', 'off' )
644   set( handles.controlPlayPassengerSignal,         'Enable', 'off' )
645   set( handles.controlPlayAmbientimeTriggerSignal, 'Enable', 'off' )
646   set( handles.uitable2,                           'Data', [] )
647
648   function handles = plotNextPreviousSamples( handles )
649   % This code is used for next, discard, and previous
650
651   % Turn off second y axis between plots
652   if isfield( handles, 'ax' )
653       if length ( handles.ax ) > 1
654           set( handles.ax( 2 ), 'Visible' , 'Off' )
655           cla( handles.ax( 2 ) )
656       end
657   end
658
659   if ~ isempty( handles.data )
660       if ~ isempty( handles.data( handles.data.opCond == handles.opCond, : ) )
661
662           % Clear all displays
663           set( handles.displayDriverSignalFile,        'String', '') ;
664           set( handles.displayDriverPreAmbientFile,    'String', '') ;
665           set( handles.displayPassengerSignalFile,     'String', '') ;
666           set( handles.displayPassengerPreAmbientFile, 'String', '') ;
667           set( handles.displayTriggerFile,             'String', '') ;
668           set( handles.displayFrontMicSignalFile,      'String', '') ;
669           set( handles.displayFrontMicPreAmbientFile,  'String', '') ;
670
671           % Enable displays according to OC
672           if strcmpi( handles.opCond, 'Stationary' );
673               set( handles.textPP,                'Enable', 'off' );
674               set( handles.textTrSig,             'Enable', 'off' );
675               set( handles.checkboxTrigger,       'Value', 0 );
676               set( handles.checkboxTrigger,       'Enable', 'off' );
677               set( handles.displayTriggerFile,    'Enable', 'off' );
678               set( handles.controlTriggerChannel, 'Enable', 'off' );
679               set( handles.textTriggerTime,       'Enable', 'off' );
680               set( handles.textTrThres,           'Enable', 'off' );
681               set( handles.textTrSec,             'Enable', 'off' );
682               set( handles.editTriggerThreshold,  'Enable', 'off' );
683               set( handles.textTrPercent,         'Enable', 'off' );
684
685               set( handles.textFront,                         'Enable', 'on' );
686               set( handles.textFrSig,                         'Enable', 'on' );
687               set( handles.displayFrontMicSignalFile,         'Enable', 'on' );
688               set( handles.controlFrontMicSignalChannel,      'Enable', 'on' );
689               set( handles.textFrPreAmb,                      'Enable', 'on' );
690               set( handles.displayFrontMicPreAmbientFile,     'Enable', 'on' );
691               set( handles.controlFrontMicPreAmbientChannel, 'Enable', 'on' );
692
```

```
693         elseif  strcmpi( handles.opCond, '10 km/h' );
694             set( handles.textPP,                 'Enable', 'on' );
695             set( handles.textTrSig,              'Enable', 'on' );
696             set( handles.checkboxTrigger,        'Value', 1 );
697             set( handles.checkboxTrigger,        'Enable', 'on' );
698             set( handles.displayTriggerFile,     'Enable', 'on' );
699             set( handles.controlTriggerChannel, 'Enable', 'on' );
700             set( handles.textTriggerTime,        'Enable', 'on' );
701             set( handles.textTrThres,            'Enable', 'on' );
702             set( handles.textTrSec,              'Enable', 'on' );
703             set( handles.editTriggerThreshold,  'Enable', 'on' );
704             set( handles.textTrPercent,          'Enable', 'on' );
705
706             set( handles.textFront,                          'Enable', 'off' );
707             set( handles.textFrSig,                          'Enable', 'off' );
708             set( handles.displayFrontMicSignalFile,          'Enable', 'off' );
709             set( handles.controlFrontMicSignalChannel,      'Enable', 'off' );
710             set( handles.textFrPreAmb,                       'Enable', 'off' );
711             set( handles.displayFrontMicPreAmbientFile,     'Enable', 'off' );
712             set( handles.controlFrontMicPreAmbientChannel, 'Enable', 'off' );
713
714         elseif  strcmpi( handles.opCond, '20 km/h' );
715             set( handles.textPP,                 'Enable', 'on' );
716             set( handles.textTrSig,              'Enable', 'on' );
717             set( handles.checkboxTrigger,        'Value', 1 );
718             set( handles.checkboxTrigger,        'Enable', 'on' );
719             set( handles.displayTriggerFile,     'Enable', 'on' );
720             set( handles.controlTriggerChannel, 'Enable', 'on' );
721             set( handles.textTriggerTime,        'Enable', 'on' );
722             set( handles.textTrThres,            'Enable', 'on' );
723             set( handles.textTrSec,              'Enable', 'on' );
724             set( handles.editTriggerThreshold,  'Enable', 'on' );
725             set( handles.textTrPercent,          'Enable', 'on' );
726
727             set( handles.textFront,                          'Enable', 'off' );
728             set( handles.textFrSig,                          'Enable', 'off' );
729             set( handles.displayFrontMicSignalFile,          'Enable', 'off' );
730             set( handles.controlFrontMicSignalChannel,      'Enable', 'off' );
731             set( handles.textFrPreAmb,                       'Enable', 'off' );
732             set( handles.displayFrontMicPreAmbientFile,     'Enable', 'off' );
733             set( handles.controlFrontMicPreAmbientChannel, 'Enable', 'off' );
734
735         elseif  strcmpi( handles.opCond, '30 km/h' );
736             set( handles.textPP,                 'Enable', 'on' );
737             set( handles.textTrSig,              'Enable', 'on' );
738             set( handles.checkboxTrigger,        'Value', 1 );
739             set( handles.checkboxTrigger,        'Enable', 'on' );
740             set( handles.displayTriggerFile,     'Enable', 'on' );
741             set( handles.controlTriggerChannel, 'Enable', 'on' );
742             set( handles.textTriggerTime,        'Enable', 'on' );
```

```
743             set( handles.textTrThres,           'Enable', 'on' );
744             set( handles.textTrSec,             'Enable', 'on' );
745             set( handles.editTriggerThreshold,  'Enable', 'on' );
746             set( handles.textTrPercent,         'Enable', 'on' );
747 
748             set( handles.textFront,                          'Enable', 'off' );
749             set( handles.textFrSig,                          'Enable', 'off' );
750             set( handles.displayFrontMicSignalFile,          'Enable', 'off' );
751             set( handles.controlFrontMicSignalChannel,       'Enable', 'off' );
752             set( handles.textFrPreAmb,                       'Enable', 'off' );
753             set( handles.displayFrontMicPreAmbientFile,      'Enable', 'off' );
754             set( handles.controlFrontMicPreAmbientChannel,   'Enable', 'off' );
755 
756         elseif  strcmpi( handles.opCond, 'Reverse' );
757             set( handles.textPP,                'Enable', 'off' );
758             set( handles.textTrSig,             'Enable', 'off' );
759             set( handles.checkboxTrigger,       'Value', 0 );
760             set( handles.checkboxTrigger,       'Enable', 'off' );
761             set( handles.displayTriggerFile,    'Enable', 'off' );
762             set( handles.controlTriggerChannel, 'Enable', 'off' );
763             set( handles.textTriggerTime,       'Enable', 'off' );
764             set( handles.textTrThres,           'Enable', 'off' );
765             set( handles.textTrSec,             'Enable', 'off' );
766             set( handles.editTriggerThreshold,  'Enable', 'off' );
767             set( handles.textTrPercent,         'Enable', 'off' );
768 
769             set( handles.textFront,                          'Enable', 'off' );
770             set( handles.textFrSig,                          'Enable', 'off' );
771             set( handles.displayFrontMicSignalFile,          'Enable', 'off' );
772             set( handles.controlFrontMicSignalChannel,       'Enable', 'off' );
773             set( handles.textFrPreAmb,                       'Enable', 'off' );
774             set( handles.displayFrontMicPreAmbientFile,      'Enable', 'off' );
775             set( handles.controlFrontMicPreAmbientChannel,   'Enable', 'off' );
776         end
777 
778         % Display current sample number of total sample numbers
779         set( handles.textCurrentSampleNumber, 'String', handles.currentSample );
780 
781         % Set the sampleID based on the current sample number
782         handles.sampleID = handles.numSamples( handles.currentSample );
783 
784         handles = loadData( handles );
785         handles = plotData( handles );
786 
787         setappdata( 0, 'data', handles.data);
788     end
789 end
790 
791 function handles = loadData( handles )
792 
```

```
opCond = handles.opCond;

set( handles.editSampleID, 'String', cellstr( handles.sampleID ) );
set( handles.editSampleID, 'Enable', 'on' );

%--------------------------------------------------------------------------
% Driver Signal
%--------------------------------------------------------------------------
handles.driverSignalPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'driver'
 & handles.data.signalType == 'signal' & ...
    handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
handles.driverSignalFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'driver'
 & handles.data.signalType == 'signal' & ...
    handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
set( handles.displayDriverSignalFile, 'String', [ handles.driverSignalPathName{ 1 } handles.driverSignalFileName{ 1 }
] );
set( handles.displayDriverSignalFile, 'Enable', 'on' );

driverChannel = double( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'driver' & ...
    handles.data.signalType == 'signal' & handles.data.sampleID == handles.sampleID, { 'channel' } ) );

[ handles.driverSignal_Amp, handles.driverSFs ] = wavread( [ handles.driverSignalPathName{ 1 } handles.
driverSignalFileName{ 1 } ] );
handles.driverSignal_Amp = handles.driverSignal_Amp( :, driverChannel );

set( handles.controlDriverSignalChannel, 'Value', driverChannel );
set( handles.controlDriverSignalChannel, 'Enable', 'off' );

%--------------------------------------------------------------------------
% Driver Pre-ambient
%--------------------------------------------------------------------------
handles.driverPreAmbientPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
'driver' & handles.data.signalType == 'preAmbient' & ...
    handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
handles.driverPreAmbientFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
'driver' & handles.data.signalType == 'preAmbient' & ...
    handles.data.sampleID == handles.sampleID, { 'fileName' } ) );

set( handles.displayDriverPreAmbientFile, 'String', [ handles.driverPreAmbientPathName{ 1 } handles.
driverPreAmbientFileName{ 1 } ] );
set( handles.displayDriverPreAmbientFile, 'Enable', 'on' );

driverPreAmbientChannel = double( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'driver' & ...
    handles.data.signalType == 'preAmbient' & handles.data.sampleID == handles.sampleID, { 'channel' } ) );

[ handles.driverPreAmbient_Amp, handles.driverAFs ] = wavread( [ handles.driverPreAmbientPathName{ 1 } handles.
driverPreAmbientFileName{ 1 } ] );
handles.driverPreAmbient_Amp = handles.driverPreAmbient_Amp( :, driverPreAmbientChannel );

set( handles.controlDriverPreAmbientChannel, 'Value', driverPreAmbientChannel );
```

```
835   set( handles.controlDriverPreAmbientChannel, 'Enable', 'off' );
836
837   %--------------------------------------------------------------------------
838   % Driver Post-ambient
839   %--------------------------------------------------------------------------
840   handles.driverPostAmbientPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
      'driver' & handles.data.signalType == 'postAmbient' & ...
841       handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
842   handles.driverPostAmbientFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
      'driver' & handles.data.signalType == 'postAmbient' & ...
843       handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
844
845   set( handles.displayDriverPostAmbientFile, 'String', [ handles.driverPostAmbientPathName{ 1 } handles.
      driverPostAmbientFileName{ 1 } ] );
846   set( handles.displayDriverPostAmbientFile, 'Enable', 'on' );
847
848   driverPostAmbientChannel = double( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'driver' & ...
849       handles.data.signalType == 'postAmbient' & handles.data.sampleID == handles.sampleID, { 'channel' } ) );
850
851   [ handles.driverPostAmbient_Amp, handles.driverAFs ] = wavread( [ handles.driverPostAmbientPathName{ 1 } handles.
      driverPostAmbientFileName{ 1 } ] );
852   handles.driverPostAmbient_Amp = handles.driverPostAmbient_Amp( :, driverPostAmbientChannel );
853
854   set( handles.controlDriverPostAmbientChannel, 'Value', driverPostAmbientChannel );
855   set( handles.controlDriverPostAmbientChannel, 'Enable', 'off' );
856
857   %--------------------------------------------------------------------------
858   % Passenger Signal
859   %--------------------------------------------------------------------------
860   handles.passengerSignalPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
      'passenger' & handles.data.signalType == 'signal' & ...
861       handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
862   handles.passengerSignalFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
      'passenger' & handles.data.signalType == 'signal' & ...
863       handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
864
865   set( handles.displayPassengerSignalFile, 'String', [ handles.passengerSignalPathName{ 1 } handles.
      passengerSignalFileName{ 1 } ] );
866   set( handles.displayPassengerSignalFile, 'Enable', 'on' );
867
868   passengerChannel = double( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'passenger' & ...
869       handles.data.signalType == 'signal' & handles.data.sampleID == handles.sampleID, { 'channel' } ) );
870
871   [ handles.passengerSignal_Amp, handles.passengerSFs ] = wavread( [ handles.passengerSignalPathName{ 1 } handles.
      passengerSignalFileName{ 1 } ] );
872   handles.passengerSignal_Amp = handles.passengerSignal_Amp( :, passengerChannel );
873
874   set( handles.controlPassengerSignalChannel, 'Value', passengerChannel );
875   set( handles.controlPassengerSignalChannel, 'Enable', 'off' );
876
```

```
%--------------------------------------------------------------------------
% Passenger Pre-ambient
%--------------------------------------------------------------------------
handles.passengerPreAmbientPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'passenger' & handles.data.signalType == 'preAmbient' & ...
    handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
handles.passengerPreAmbientFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'passenger' & handles.data.signalType == 'preAmbient' & ...
    handles.data.sampleID == handles.sampleID, { 'fileName' } ) );

set( handles.displayPassengerPreAmbientFile, 'String', [ handles.passengerPreAmbientPathName{ 1 } handles.passengerPreAmbientFileName{ 1 } ] );
set( handles.displayPassengerPreAmbientFile, 'Enable', 'on' );

passengerPreAmbientChannel = double( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'passenger' & ...
    handles.data.signalType == 'preAmbient' & handles.data.sampleID == handles.sampleID, { 'channel' } ) );

[ handles.passengerPreAmbient_Amp, handles.passengerAFs ] = wavread( [ handles.passengerPreAmbientPathName{ 1 } handles.passengerPreAmbientFileName{ 1 } ] );
handles.passengerPreAmbient_Amp = handles.passengerPreAmbient_Amp( :, passengerPreAmbientChannel );

set( handles.controlPassengerPreAmbientChannel, 'Value', passengerPreAmbientChannel );
set( handles.controlPassengerPreAmbientChannel, 'Enable', 'off' );

%--------------------------------------------------------------------------
% Passenger Post-ambient
%--------------------------------------------------------------------------
handles.passengerPostAmbientPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'passenger' & handles.data.signalType == 'postAmbient' & ...
    handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
handles.passengerPostAmbientFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'passenger' & handles.data.signalType == 'postAmbient' & ...
    handles.data.sampleID == handles.sampleID, { 'fileName' } ) );

set( handles.displayPassengerPostAmbientFile, 'String', [ handles.passengerPostAmbientPathName{ 1 } handles.passengerPostAmbientFileName{ 1 } ] );
set( handles.displayPassengerPostAmbientFile, 'Enable', 'on' );

passengerPostAmbientChannel = double( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'passenger' & ...
    handles.data.signalType == 'postAmbient' & handles.data.sampleID == handles.sampleID, { 'channel' } ) );

[ handles.passengerPostAmbient_Amp, handles.passengerAFs ] = wavread( [ handles.passengerPostAmbientPathName{ 1 } handles.passengerPostAmbientFileName{ 1 } ] );
handles.passengerPostAmbient_Amp = handles.passengerPostAmbient_Amp( :, passengerPostAmbientChannel );

set( handles.controlPassengerPostAmbientChannel, 'Value', passengerPostAmbientChannel);
set( handles.controlPassengerPostAmbientChannel, 'Enable', 'off' );

```

```
917   %--------------------------------------------------------------------------
918   % Front Signal
919   %--------------------------------------------------------------------------
920   if strcmp( opCond, 'Stationary' ) == 1
921       set( handles.textFront, 'Enable', 'on' );
922       set( handles.textFrSig, 'Enable', 'on' );
923       set( handles.textFrPreAmb, 'Enable', 'on' );
924       handles.frontimeTriggerSignalPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.
      micSide == 'front' & handles.data.signalType == 'signal' & ...
925           handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
926       handles.frontimeTriggerSignalFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.
      micSide == 'front' & handles.data.signalType == 'signal' & ...
927           handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
928       set( handles.displayFrontMicSignalFile, 'String', [ handles.frontimeTriggerSignalPathName{ 1 } handles.
      frontimeTriggerSignalFileName{ 1 } ] );
929       set( handles.displayFrontMicSignalFile, 'Enable', 'on' );
930
931       frontChannel = double( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'front' & ...
932           handles.data.signalType == 'signal' & handles.data.sampleID == handles.sampleID, { 'channel' } ) );
933
934       [ handles.frontSignal_Amp, handles.frontSFs ] = wavread( [ handles.frontimeTriggerSignalPathName{ 1 } handles.
      frontimeTriggerSignalFileName{ 1 } ] );
935       handles.frontSignal_Amp = handles.frontSignal_Amp( :, frontChannel );
936
937       set( handles.controlFrontMicSignalChannel, 'Value', frontChannel );
938       set( handles.controlFrontMicSignalChannel, 'Enable', 'off' );
939
940       %----------------------------------------------------------------------
941       % Front Pre-ambient
942       %----------------------------------------------------------------------
943       handles.frontPreAmbientPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
      'front' & handles.data.signalType == 'preAmbient' & ...
944           handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
945       handles.frontPreAmbientFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
      'front' & handles.data.signalType == 'preAmbient' & ...
946           handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
947       set( handles.displayFrontMicPreAmbientFile, 'String', [ handles.frontPreAmbientPathName{ 1 } handles.
      frontPreAmbientFileName{ 1 } ] );
948       set( handles.displayFrontMicPreAmbientFile, 'Enable', 'on' );
949
950       frontPreAmbientChannel = double( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'front' &
      ...
951           handles.data.signalType == 'preAmbient' & handles.data.sampleID == handles.sampleID, { 'channel' } ) );
952
953       [ handles.frontPreAmbient_Amp, handles.frontAFs ] = wavread( [ handles.frontPreAmbientPathName{ 1 } handles.
      frontPreAmbientFileName{ 1 } ] );
954       handles.frontPreAmbient_Amp = handles.frontPreAmbient_Amp( :, frontPreAmbientChannel );
955
956       set( handles.controlFrontMicPreAmbientChannel, 'Value', frontPreAmbientChannel );
957       set( handles.controlFrontMicPreAmbientChannel, 'Enable', 'off' );
```

```
958
959         %-------------------------------------------------------------------------
960         % Front Post-ambient
961         %-------------------------------------------------------------------------
962         handles.frontPostAmbientPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
            'front' & handles.data.signalType == 'postAmbient' & ...
963             handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
964         handles.frontPostAmbientFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
            'front' & handles.data.signalType == 'postAmbient' & ...
965             handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
966         set( handles.displayFrontMicPostAmbientFile, 'String', [ handles.frontPostAmbientPathName{ 1 } handles.
            frontPostAmbientFileName{ 1 } ] );
967         set( handles.displayFrontMicPostAmbientFile, 'Enable', 'on' );
968
969         frontPostAmbientChannel = double( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'front' &
            ...
970             handles.data.signalType == 'postAmbient' & handles.data.sampleID == handles.sampleID, { 'channel' } ) );
971
972         [ handles.frontPostAmbient_Amp, handles.frontAFs ] = wavread( [ handles.frontPostAmbientPathName{ 1 } handles.
            frontPostAmbientFileName{ 1 } ] );
973         handles.frontPostAmbient_Amp = handles.frontPostAmbient_Amp( :, frontPostAmbientChannel );
974
975         set( handles.controlFrontMicPostAmbientChannel, 'Value', frontPostAmbientChannel );
976         set( handles.controlFrontMicPostAmbientChannel, 'Enable', 'off' );
977
978     end
979
980     %-------------------------------------------------------------------------
981     % Trigger
982     %-------------------------------------------------------------------------
983     if ~ isempty( handles.data( handles.data.opCond == opCond & handles.data.signalType == 'trigger' & handles.data.
        sampleID == handles.sampleID, : ) )
984         handles.triggerPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.signalType ==
            'trigger' & handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
985         handles.triggerFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.signalType ==
            'trigger' & handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
986         set( handles.displayTriggerFile, 'String', [ handles.triggerPathName{ 1 } handles.triggerFileName{ 1 } ] );
987         set( handles.displayTriggerFile, 'Enable', 'on' );
988
989         triggerChannel = double( handles.data( handles.data.opCond == opCond & handles.data.signalType == 'trigger' & ...
990             handles.data.sampleID == handles.sampleID, { 'channel' } ) );
991
992         [ handles.triggerSignal_Amp, handles.triggerFs ] = wavread( [ handles.triggerPathName{ 1 } handles.triggerFileName
            { 1 } ] );
993         handles.triggerSignal_Amp = handles.triggerSignal_Amp( :, triggerChannel );
994
995         set( handles.controlTriggerChannel, 'Value', triggerChannel);
996         set( handles.controlTriggerChannel, 'Enable', 'off' );
997
998
```

```
999         tThresh1 = handles.data( handles.data.opCond == opCond & handles.data.signalType == 'trigger' & handles.data.
            sampleID == handles.sampleID, { 'TriggerThreshold' } );
1000        tThresh2 = tThresh1.TriggerThreshold ;
1001        tThresh3 = cell2mat( tThresh2 ) ;
1002
1003        set( handles.editTriggerThreshold, 'String', tThresh3 );
1004
1005        set( handles.checkboxTrigger,      'Value', 1 );
1006        set( handles.checkboxTrigger,      'Enable', 'on' );
1007        set( handles.editTriggerThreshold, 'Enable', 'on' );
1008    end
1009
1010    function handles = plotData( handles )
1011
1012    % Create time signals for all signals
1013    handles.driverSignal_Time = ( 0 : length( handles.driverSignal_Amp ) - 1 )/handles.driverSFs;%SFs = signal, Fs
1014    handles.driverPreAmbient_Time = ( 0 : length( handles.driverPreAmbient_Amp ) - 1 )/handles.driverAFs;% AFs = ambient,
        Fs
1015    handles.driverPostAmbient_Time = ( 0 : length( handles.driverPostAmbient_Amp ) - 1 )/handles.driverAFs;
1016    handles.passengerSignal_Time = ( 0 : length( handles.passengerSignal_Amp ) - 1 )/handles.passengerSFs;
1017    handles.passengerPreAmbient_Time = ( 0 : length( handles.passengerPreAmbient_Amp ) -1 )/handles.passengerAFs;
1018    handles.passengerPostAmbient_Time = ( 0 : length( handles.passengerPostAmbient_Amp ) -1 )/handles.passengerAFs;
1019
1020    if strcmp( handles.opCond, 'Stationary' ) == 1
1021        handles.frontSignal_Time = ( 0 : length( handles.frontSignal_Amp ) - 1 )/handles.frontSFs;
1022        handles.frontPreAmbient_Time = ( 0 : length( handles.frontPreAmbient_Amp ) -1 )/handles.frontAFs;
1023        handles.frontPostAmbient_Time = ( 0 : length( handles.frontPostAmbient_Amp ) -1 )/handles.frontAFs;
1024
1025    end
1026
1027    if strcmp (handles.opCond,'10 km/h' ) || strcmp (handles.opCond,'20 km/h' )||  strcmp (handles.opCond,'30 km/h' );
1028        if handles.checkboxTrigger
1029            handles.triggerSignal_Time = ( 0 : length( handles.triggerSignal_Amp ) - 1 )/handles.triggerFs;
1030        else % If trigger doesn't exist, create one that uses the whole signal --> trigger checkbox is always checked, so
            this shouldn't be needed.
1031            handles.triggerSignal_Amp =  0 * handles.driverSignal_Time;
1032            handles.triggerSignal_Amp( end ) = 1;
1033            handles.triggerSignal_Time = handles.driverSignal_Time;
1034        end
1035    end
1036
1037    cla( handles.axesDisplay )
1038    axes( handles.axesDisplay ) % Need to explicitly set this or it can change
1039
1040    % Clear old signals
1041    if isfield( handles, 'ax' )
1042        if length ( handles.ax ) > 1
1043            set( handles.ax( 2 ), 'Visible' , 'Off' )
1044            cla( handles.ax( 2 ) )
1045        end
```

```
1046    else
1047        handles.ax = gca ;
1048    end
1049
1050    % Plot new signals
1051    if isfield( handles, 'driverSignal_Amp' )
1052
1053        handles.hPlotimeTriggerSignal_driver = ...
1054            plot( handles.axesDisplay, handles.driverSignal_Time, handles.driverSignal_Amp, 'color', [0,0,1] );
1055
1056        % Yaxis label
1057        set( get( handles.ax( 1 ), 'ylabel' ), 'String', 'Acoustic Pressure (Pa)' ) ;
1058
1059        pause( 1 ) ;
1060
1061        if isfield ( handles, 'hAxis' )
1062            set(handles.hAxis(2), 'Visible', 'off') ;
1063        end
1064
1065        hold on
1066
1067        handles.hPlotPreAmbient_driver = ...
1068            plot( handles.axesDisplay, handles.driverPreAmbient_Time, handles.driverPreAmbient_Amp, 'color', [0,0.6,1] );
1069        pause( 1 ) ;
1070
1071        handles.hPlotPostAmbient_driver = ...
1072            plot( handles.axesDisplay, handles.driverPostAmbient_Time, handles.driverPostAmbient_Amp, 'color', [0,1,1] );
1073        pause( 1 ) ;
1074
1075    end
1076
1077    if isfield( handles, 'passengerSignal_Amp' )
1078        handles.hPlotimeTriggerSignal_passenger = ...
1079            plot( handles.axesDisplay, handles.passengerSignal_Time, handles.passengerSignal_Amp, 'color', [1,0,0] );
1080        pause( 1 ) ;
1081
1082        handles.hPlotPreAmbient_passenger = ...
1083            plot( handles.axesDisplay, handles.passengerPreAmbient_Time, handles.passengerPreAmbient_Amp, 'color', [1,0,
            0.4]  );
1084        pause( 1 ) ;
1085
1086        handles.hPlotPostAmbient_passenger = ...
1087            plot( handles.axesDisplay, handles.passengerPostAmbient_Time, handles.passengerPostAmbient_Amp, 'color', [1,0,
            1]  );
1088        pause( 1 ) ;
1089
1090    end
1091
1092    if strcmp ( handles.opCond,'Stationary' )
```

```
1093        if isfield( handles, 'frontSignal_Amp' )
1094            handles.hPlotimeTriggerSignal_front = ...
1095                plot( handles.axesDisplay, handles.frontSignal_Time, handles.frontSignal_Amp, 'color', [0,0,0] );
1096            pause( 1 ) ;
1097
1098            handles.hPlotPreAmbient_front = ...
1099                plot( handles.axesDisplay, handles.frontPreAmbient_Time, handles.frontPreAmbient_Amp, 'color', [0.3,0.3,
                    0.3]);
1100            pause( 1 ) ;
1101
1102            handles.hPlotPostAmbient_front = ...
1103                plot( handles.axesDisplay, handles.frontPostAmbient_Time, handles.frontPostAmbient_Amp, 'color', [0.6,0.6,
                    0.6] );
1104            pause( 1 ) ;
1105        end
1106    end
1107
1108    if strcmp (handles.opCond,'10 km/h' ) || strcmp (handles.opCond,'20 km/h' )||  strcmp (handles.opCond,'30 km/h' );
1109        if handles.checkboxTrigger
1110            % Prep trigger percentage plot
1111            [ handles.MaxTriggerSignal, handles.MaxTriggerSignalIdx ] = max( abs( handles.triggerSignal_Amp ) );
1112            handles.timeAtMaxTrigger = handles.triggerSignal_Time( handles.MaxTriggerSignalIdx );
1113            handles.triggerScaledTo100 = ( handles.triggerSignal_Amp/handles.MaxTriggerSignal ) * 100;
1114            handles.triggerThresholdValue = str2num( get( handles.editTriggerThreshold, 'String' ) );
1115            handles.idxAtTriggerThreshold = find( (handles.triggerScaledTo100) > handles.triggerThresholdValue, 1 );
1116            handles.timeAtTriggerThreshold = handles.triggerSignal_Time( handles.idxAtTriggerThreshold );
1117
1118            % Update trigger time display
1119            set( handles.editTrigger, 'String', handles.timeAtTriggerThreshold )
1120
1121            % Plot trigger signal
1122            if length ( handles.ax ) > 1
1123                set( handles.ax( 2 ), 'Visible' , 'On' )
1124                axes( handles.ax( 2 ) ) ;
1125                hold on
1126                handles.hPlotTrigger = plot( handles.ax( 2 ), handles.triggerSignal_Time, handles.triggerScaledTo100, 'g'
                    ,'LineWidth', 2);
1127
1128            else
1129                ax1hv = get( handles.ax( 1 ), 'HandleVisibility' ) ;
1130                handles.ax( 2 ) = axes( 'HandleVisibility', ax1hv, 'Units', get( handles.ax( 1 ), 'Units' ), ...
1131                    'Position',get( handles.ax( 1 ), 'Position' ), 'Parent', get( handles.ax( 1 ),'Parent' ) );
1132
1133                handles.hPlotTrigger = plot( handles.ax( 2 ), handles.triggerSignal_Time, handles.triggerScaledTo100, 'g'
                    ,'LineWidth', 2);
1134
1135                set( handles.ax( 2 ),'YAxisLocation','right','Color','none', ...
1136                    'XGrid','off','YGrid','off','Box','off', ...
1137                    'HitTest','off');
1138                xlim2 = get( handles.ax( 2 ),'XLim' );
```

```
1139                ylim2 = get( handles.ax( 2 ), 'YLim' );
1140
1141                set( handles.ax( 2 ), 'Xtick', [] )
1142                set( handles.ax( 2 ), 'YLim', [0 100 ] )
1143                set( handles.ax( 2 ), 'Ytick', [0:10:100] )
1144                set( handles.ax( 2 ), 'YTickLabel', [0:10:100] )
1145
1146                % Fix extra tick marks on 2nd y axis
1147                linkaxes( handles.ax, 'x' );
1148                set( handles.ax( 1 ), 'Box', 'off' )
1149                set( handles.ax( 2 ), 'Box', 'off' )
1150                saveXTick = get( handles.ax( 1 ), 'XTick' ) ;
1151                set( handles.ax( 2 ), 'XTick', saveXTick, 'XTickLabel', '', 'XAxisLocation', 'Top' )
1152
1153            end
1154
1155            % Plot trigger threshold line
1156            handles.trigX1 = min( handles.triggerSignal_Time );
1157            handles.trigX2 = max( handles.triggerSignal_Time );
1158
1159            hold on
1160            handles.hPlotTriggerPercent = plot( handles.ax( 2 ), [ handles.trigX1 handles.trigX2 ], [handles.
                triggerThresholdValue handles.triggerThresholdValue], 'color', [0 0.5 0], 'LineStyle', '- -', 'LineWidth', 3) ;
1161            hold off
1162
1163        end
1164    end
1165
1166    if strcmp (handles.opCond,'10 km/h' ) || strcmp (handles.opCond,'20 km/h' )||  strcmp (handles.opCond,'30 km/h' );
1167        set( get( handles.ax( 2 ), 'ylabel' ), 'String', 'Percent of Max Trigger Level (%)' ) ;
1168    else
1169        minSignal = min( [ min( handles.driverSignal_Time ) , min( handles.driverPreAmbient_Time ), min( handles.
            driverPostAmbient_Time ), min( handles.passengerSignal_Time ), min( handles.passengerPreAmbient_Time ), min(
            handles.passengerPostAmbient_Time )]) ;
1170        maxSignal = max( [ max( handles.driverSignal_Time ) , max( handles.driverPreAmbient_Time ), max( handles.
            driverPostAmbient_Time ), max( handles.passengerSignal_Time ), max( handles.passengerPreAmbient_Time ), max(
            handles.passengerPostAmbient_Time )]) ;
1171        axis( [ minSignal maxSignal -inf inf ] )
1172    end
1173
1174    % xlabel( 'Time, seconds' )
1175    set( get( handles.ax( 1 ), 'ylabel' ), 'String', 'Acoustic Pressure (Pa)' ) ;
1176
1177    % title('Determine Analysis Region')
1178    set( handles.axesDisplay, 'XScale', 'linear' )
1179
1180    % --- Executes on button press in pushbuttonClearAll.
1181    function pushbuttonClearAll_Callback(hObject, eventdata, handles)
1182    % hObject    handle to pushbuttonClearAll (see GCBO)
1183    % eventdata  reserved - to be defined in a future version of MATLAB
```

```
1184   % handles    structure with handles and user data (see GUIDATA)
1185
1186   % Clear data
1187   handles.data = [] ;
1188   handles.samplesDone = [] ;
1189   handles.numSamples = [] ;
1190   handles.sampleID = [] ;
1191   handles.driverSignalPathName = [] ;
1192   handles.driverSignalFileName = [] ;
1193   handles.driverSignal_Amp = [] ;
1194   handles.driverPreAmbientPathName = [] ;
1195   handles.driverPreAmbientFileName = [] ;
1196   handles.driverPreAmbient_Amp = [] ;
1197   handles.driverPostAmbientPathName = [] ;
1198   handles.driverPostAmbientFileName = [] ;
1199   handles.driverPostAmbient_Amp = [] ;
1200   handles.passengerSignalPathName = [] ;
1201   handles.passengerSignalFileName = [] ;
1202   handles.passengerSignal_Amp = [] ;
1203   handles.passengerPreAmbientPathName = [] ;
1204   handles.passengerPreAmbientFileName = [] ;
1205   handles.passengerPreAmbient_Amp = [] ;
1206   handles.passengerPostAmbientPathName = [] ;
1207   handles.passengerPostAmbientFileName = [] ;
1208   handles.passengerPostAmbient_Amp = [] ;
1209   handles.triggerPathName = [] ;
1210   handles.triggerFileName = [] ;
1211   handles.triggerSignal_Amp = [] ;
1212   handles.timeAtMaxTrigger = [] ;
1213   handles.triggerScaledTo100 = [] ;
1214   handles.triggerThresholdValue = [] ;
1215   handles.idxAtTriggerThreshold = [] ;
1216   handles.timeAtTriggerThreshold = [] ;
1217   handles.MaxTriggerSignal = [] ;
1218   handles.MaxTriggerSignalIdx = [] ;
1219   handles.hPlotimeTriggerSignal_driver = [] ;
1220   handles.hPlotPreAmbient_driver = [] ;
1221   handles.hPlotPostAmbient_driver = [] ;
1222   handles.hPlotimeTriggerSignal_passenger = [] ;
1223   handles.hPlotPreAmbient_passenger = [] ;
1224   handles.hPlotPostAmbient_passenger = [] ;
1225
1226   guidata( hObject, handles ) ; % Update handles
1227
1228   % Clear Displays (Vehicle Information, User Comments)
1229   set( handles.editYear,                      'String', 'Edit Text') ;
1230   set( handles.editMake,                      'String', 'Edit Text') ;
1231   set( handles.editModel,                     'String', 'Edit Text') ;
1232   set( handles.editNHTSANumber,               'String', 'Edit Text') ;
1233   set( handles.editVIN,                       'String', 'Edit Text') ;
```

```
1234   set( handles.editUserComments,                'String', 'Edit Text') ;
1235
1236   % Clear Displays (Data Review)
1237   set( handles.editSampleID,                      'String', ' NA') ;
1238   set( handles.textTotalSamplesafterDiscard,      'String', ' NA') ;
1239   set( handles.textCurrentSampleNumber,           'String', ' NA') ;
1240   set( handles.textTotalSampleNumbers,            'String', ' NA') ;
1241
1242   set( handles.controlOperatingCondition,         'Value', 1 ) ;
1243   set( handles.displayDriverSignalFile,           'String', ' NA') ;
1244   set( handles.controlDriverSignalChannel,        'Value', 1 );
1245   set( handles.displayDriverPreAmbientFile,       'String', ' NA') ;
1246   set( handles.controlDriverPreAmbientChannel,    'Value', 1 );
1247   set( handles.displayDriverPostAmbientFile,      'String', ' NA') ;
1248   set( handles.controlDriverPostAmbientChannel,   'Value', 1 );
1249   set( handles.displayPassengerSignalFile,        'String', ' NA') ;
1250   set( handles.controlPassengerSignalChannel,     'Value', 2 );
1251   set( handles.displayPassengerPreAmbientFile,    'String', ' NA') ;
1252   set( handles.controlPassengerPreAmbientChannel, 'Value', 2 );
1253   set( handles.displayPassengerPostAmbientFile,   'String', ' NA') ;
1254   set( handles.controlPassengerPostAmbientChannel,'Value', 2 );
1255   set( handles.displayFrontMicSignalFile,         'String', ' NA') ;
1256   set( handles.controlFrontMicSignalChannel,      'Value', 3 );
1257   set( handles.displayFrontMicPreAmbientFile,     'String', ' NA') ;
1258   set( handles.controlFrontMicPreAmbientChannel,  'Value', 3 );
1259   set( handles.displayFrontMicPostAmbientFile ,   'String', ' NA') ;
1260   set( handles.controlFrontMicPostAmbientChannel, 'Value', 3 );
1261   set( handles.displayTriggerFile,                'String', ' NA') ;
1262   set( handles.controlTriggerChannel,             'Value', 3 );
1263   set( handles.editTrigger,                       'String', ' Edit Text') ;
1264
1265   if isfield( handles, 'ax' )
1266       if length ( handles.ax ) > 1
1267           set( handles.ax( 2 ), 'Visible' , 'Off' )
1268           cla( handles.ax( 2 ) )
1269       end
1270   end
1271
1272   cla( handles.axesDisplay )
1273   axes( handles.axesDisplay )
1274
1275   % Clear Displays & Adjust Visibility (Compute and Resutls)
1276   set( handles.displayRunDurationValue,     'String', '-' );
1277   set( handles.textProcessingStep,          'String', 'Processing sample' );
1278   set( handles.displayRunStatusCurrentFile, 'String', '0' );
1279   set( handles.displayRunStatusTotalFiles,  'String', '0' );
1280   set( handles.displayCurrentOC,            'String', ' NA' );
1281
1282   set( handles.display2dBError,             'Visible', 'off' );
1283   set( handles.displayAnalysisComplete,     'Visible', 'off' );
```

```
1284   set( handles.textAnalysisStopped,          'Visible', 'off' );
1285   set( handles.textRunStatusOf,              'Visible', 'on' );
1286   set( handles.displayRunStatusCurrentFile, 'Visible', 'on' );
1287   set( handles.displayRunStatusTotalFiles,  'Visible', 'on' );
1288   set( handles.displayCurrentOC,             'Visible', 'on' );
1289   set( handles.textRunStatusFor,             'Visible', 'on' );
1290
1291   set( handles.displayOutputFilePath,        'String', ' NA'  );
1292   set( handles.uitableResults,               'Data', [] )
1293   set( handles.displayCompliance,            'String', ' NA') ;
1294
1295   % Clear Saved Data For Tables In GUI_fileLoad2
1296   setappdata( 0, 'loadedTableDataFlag', 0 ) ;
1297   setappdata( 0, 'loadedTableDataStationary', [] ) ;
1298   setappdata( 0, 'loadedTableData10kmh', [] ) ;
1299   setappdata( 0, 'loadedTableData20kmh', [] ) ;
1300   setappdata( 0, 'loadedTableData30kmh', [] ) ;
1301   setappdata( 0, 'loadedTableDataReverse', [] ) ;
1302
1303
1304
1305   %**************************************************************************
1306   %**************************************************************************
1307   %      Create Functions
1308   %**************************************************************************
1309   %**************************************************************************
1310
1311   % --- Executes during object creation, after setting all properties.
1312   function radioFileLoad_CreateFcn(hObject, eventdata, handles)
1313
1314   % --- Executes during object creation, after setting all properties.
1315   function radioBatchLoad_CreateFcn(hObject, eventdata, handles)
1316
1317   % --- Executes during object creation, after setting all properties.
1318   function controlFileSelect_CreateFcn(hObject, eventdata, handles)
1319   % hObject    handle to controlFileSelect (see GCBO)
1320   % eventdata  reserved - to be defined in a future version of MATLAB
1321   % handles    empty - handles not created until after all CreateFcns called
1322
1323   % --- Executes during object creation, after setting all properties.
1324   function editYear_CreateFcn(hObject, eventdata, handles)
1325   % hObject    handle to editYear (see GCBO)
1326   % eventdata  reserved - to be defined in a future version of MATLAB
1327   % handles    empty - handles not created until after all CreateFcns called
1328
1329   % Hint: edit controls usually have a white background on Windows.
1330   %       See ISPC and COMPUTER.
1331   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1332       set(hObject,'BackgroundColor','white');
1333   end
```

```
1334
1335  % --- Executes during object creation, after setting all properties.
1336  function editMake_CreateFcn(hObject, eventdata, handles)
1337  % hObject    handle to editMake (see GCBO)
1338  % eventdata  reserved - to be defined in a future version of MATLAB
1339  % handles    empty - handles not created until after all CreateFcns called
1340
1341  % Hint: edit controls usually have a white background on Windows.
1342  %       See ISPC and COMPUTER.
1343  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1344      set(hObject,'BackgroundColor','white');
1345  end
1346
1347  % --- Executes during object creation, after setting all properties.
1348  function editModel_CreateFcn(hObject, eventdata, handles)
1349  % hObject    handle to editModel (see GCBO)
1350  % eventdata  reserved - to be defined in a future version of MATLAB
1351  % handles    empty - handles not created until after all CreateFcns called
1352
1353  % Hint: edit controls usually have a white background on Windows.
1354  %       See ISPC and COMPUTER.
1355  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1356      set(hObject,'BackgroundColor','white');
1357  end
1358
1359  % --- Executes during object creation, after setting all properties.
1360  function editNHTSANumber_CreateFcn(hObject, eventdata, handles)
1361  % hObject    handle to editNHTSANumber (see GCBO)
1362  % eventdata  reserved - to be defined in a future version of MATLAB
1363  % handles    empty - handles not created until after all CreateFcns called
1364
1365  % Hint: edit controls usually have a white background on Windows.
1366  %       See ISPC and COMPUTER.
1367  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1368      set(hObject,'BackgroundColor','white');
1369  end
1370
1371  % --- Executes during object creation, after setting all properties.
1372  function editVIN_CreateFcn(hObject, eventdata, handles)
1373  % hObject    handle to editVIN (see GCBO)
1374  % eventdata  reserved - to be defined in a future version of MATLAB
1375  % handles    empty - handles not created until after all CreateFcns called
1376
1377  % Hint: edit controls usually have a white background on Windows.
1378  %       See ISPC and COMPUTER.
1379  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1380      set(hObject,'BackgroundColor','white');
1381  end
1382
1383  % --- Executes during object creation, after setting all properties.
```

```
function editUserComments_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editUserComments (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlOperatingCondition_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayDriverSignalFile_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlDriverSignalChannel_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayDriverPreAmbientFile_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlDriverPreAmbientChannel_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayDriverPostAmbientFile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to displayDriverPostAmbientFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlDriverPostAmbientChannel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to controlDriverPostAmbientChannel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayPassengerSignalFile_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlPassengerSignalChannel_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayPassengerPreAmbientFile_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlPassengerPreAmbientChannel_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayPassengerPostAmbientFile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to displayPassengerPostAmbientFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
1484
1485  % --- Executes during object creation, after setting all properties.
1486  function controlPassengerPostAmbientChannel_CreateFcn(hObject, eventdata, handles)
1487  % hObject    handle to controlPassengerPostAmbientChannel (see GCBO)
1488  % eventdata  reserved - to be defined in a future version of MATLAB
1489  % handles    empty - handles not created until after all CreateFcns called
1490
1491  % Hint: popupmenu controls usually have a white background on Windows.
1492  %       See ISPC and COMPUTER.
1493  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1494      set(hObject,'BackgroundColor','white');
1495  end
1496
1497  % --- Executes during object creation, after setting all properties.
1498  function displayFrontMicSignalFile_CreateFcn(hObject, eventdata, handles)
1499  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1500      set(hObject,'BackgroundColor','white');
1501  end
1502
1503  % --- Executes during object creation, after setting all properties.
1504  function controlFrontMicSignalChannel_CreateFcn(hObject, eventdata, handles)
1505  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1506      set(hObject,'BackgroundColor','white');
1507  end
1508
1509  % --- Executes during object creation, after setting all properties.
1510  function displayFrontMicPreAmbientFile_CreateFcn(hObject, eventdata, handles)
1511  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1512      set(hObject,'BackgroundColor','white');
1513  end
1514
1515  % --- Executes during object creation, after setting all properties.
1516  function controlFrontMicPreAmbientChannel_CreateFcn(hObject, eventdata, handles)
1517  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1518      set(hObject,'BackgroundColor','white');
1519  end
1520
1521  % --- Executes during object creation, after setting all properties.
1522  function displayFrontMicPostAmbientFile_CreateFcn(hObject, eventdata, handles)
1523  % hObject    handle to displayFrontMicPostAmbientFile (see GCBO)
1524  % eventdata  reserved - to be defined in a future version of MATLAB
1525  % handles    empty - handles not created until after all CreateFcns called
1526
1527  % Hint: edit controls usually have a white background on Windows.
1528  %       See ISPC and COMPUTER.
1529  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
1530      set(hObject,'BackgroundColor','white');
1531  end
1532
1533  % --- Executes during object creation, after setting all properties.
```

```
function controlFrontMicPostAmbientChannel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to controlFrontMicPostAmbientChannel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function displayTriggerFile_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function controlTriggerChannel_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function editTrigger_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function editTriggerThreshold_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function pushbuttonCompute_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonCompute (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function displayOutputFilePath_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% % --- Executes during object creation, after setting all properties.
function displayCompliance_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
1584        set(hObject,'BackgroundColor',[0.941, 0.941, 0.941]);
1585    end
1586
1587
1588    %**************************************************************************
1589    %**************************************************************************
1590    %      Empty Callbacks / Not Currently In Use
1591    %**************************************************************************
1592    %**************************************************************************
1593
1594    function editYear_Callback(hObject, eventdata, handles)
1595    % hObject    handle to editYear (see GCBO)
1596    % eventdata  reserved - to be defined in a future version of MATLAB
1597    % handles    structure with handles and user data (see GUIDATA)
1598
1599    function editMake_Callback(hObject, eventdata, handles)
1600    % hObject    handle to editMake (see GCBO)
1601    % eventdata  reserved - to be defined in a future version of MATLAB
1602    % handles    structure with handles and user data (see GUIDATA)
1603
1604    function editModel_Callback(hObject, eventdata, handles)
1605    % hObject    handle to editModel (see GCBO)
1606    % eventdata  reserved - to be defined in a future version of MATLAB
1607    % handles    structure with handles and user data (see GUIDATA)
1608
1609    function editNHTSANumber_Callback(hObject, eventdata, handles)
1610    % hObject    handle to editNHTSANumber (see GCBO)
1611    % eventdata  reserved - to be defined in a future version of MATLAB
1612    % handles    structure with handles and user data (see GUIDATA)
1613
1614    function editVIN_Callback(hObject, eventdata, handles)
1615    % hObject    handle to editVIN (see GCBO)
1616    % eventdata  reserved - to be defined in a future version of MATLAB
1617    % handles    structure with handles and user data (see GUIDATA)
1618
1619    function editUserComments_Callback(hObject, eventdata, handles)
1620    % hObject    handle to editUserComments (see GCBO)
1621    % eventdata  reserved - to be defined in a future version of MATLAB
1622    % handles    structure with handles and user data (see GUIDATA)
1623
1624    function displayDriverSignalFile_Callback(hObject, eventdata, handles)
1625    % hObject    handle to displayDriverSignalFile (see GCBO)
1626    % eventdata  reserved - to be defined in a future version of MATLAB
1627    % handles    structure with handles and user data (see GUIDATA)
1628
1629    function controlDriverSignalChannel_Callback(hObject, eventdata, handles)
1630    % hObject    handle to controlDriverSignalChannel (see GCBO)
1631    % eventdata  reserved - to be defined in a future version of MATLAB
1632    % handles    structure with handles and user data (see GUIDATA)
1633
```

```
function displayDriverPreAmbientFile_Callback(hObject, eventdata, handles)
% hObject    handle to displayDriverPreAmbientFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on selection change in controlDriverPreAmbientChannel.
function controlDriverPreAmbientChannel_Callback(hObject, eventdata, handles) % Channels are now read-only in the
Data Review Section

if isfield( handles, 'hPlotAmbient' )

    load( [ handles.ambientPathName handles.ambientFileName ] )

    handles.ambientChannelNumber = get( handles.controlDriverPreAmbientChannel, 'Value' );
    ambientEval = sprintf( 'handles.ambient = Channel_%1.0f_Data;', handles.ambientChannelNumber );
    eval( ambientEval )

    handles = plotData( handles );

    % Update handles structure
    guidata( hObject, handles );

end

function displayDriverPostAmbientFile_Callback(hObject, eventdata, handles)
% hObject    handle to displayDriverPostAmbientFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function controlDriverPostAmbientChannel_Callback(hObject, eventdata, handles)
% hObject    handle to controlDriverPostAmbientChannel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function displayPassengerSignalFile_Callback(hObject, eventdata, handles)
% hObject    handle to displayPassengerSignalFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function controlPassengerSignalChannel_Callback(hObject, eventdata, handles)
% hObject    handle to controlPassengerSignalChannel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function displayPassengerPreAmbientFile_Callback(hObject, eventdata, handles)
% hObject    handle to displayPassengerPreAmbientFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function controlPassengerPreAmbientChannel_Callback(hObject, eventdata, handles)
```

```
1683  % hObject    handle to controlPassengerPreAmbientChannel (see GCBO)
1684  % eventdata  reserved - to be defined in a future version of MATLAB
1685  % handles    structure with handles and user data (see GUIDATA)
1686
1687  function displayPassengerPostAmbientFile_Callback(hObject, eventdata, handles)
1688  % hObject    handle to displayPassengerPostAmbientFile (see GCBO)
1689  % eventdata  reserved - to be defined in a future version of MATLAB
1690  % handles    structure with handles and user data (see GUIDATA)
1691
1692  function controlPassengerPostAmbientChannel_Callback(hObject, eventdata, handles)
1693  % hObject    handle to controlPassengerPostAmbientChannel (see GCBO)
1694  % eventdata  reserved - to be defined in a future version of MATLAB
1695  % handles    structure with handles and user data (see GUIDATA)
1696
1697  function displayFrontMicSignalFile_Callback(hObject, eventdata, handles)
1698  % hObject    handle to displayFrontMicSignalFile (see GCBO)
1699  % eventdata  reserved - to be defined in a future version of MATLAB
1700  % handles    structure with handles and user data (see GUIDATA)
1701
1702  function controlFrontMicSignalChannel_Callback(hObject, eventdata, handles)
1703  % hObject    handle to controlFrontMicSignalChannel (see GCBO)
1704  % eventdata  reserved - to be defined in a future version of MATLAB
1705  % handles    structure with handles and user data (see GUIDATA)
1706
1707  function displayFrontMicPreAmbientFile_Callback(hObject, eventdata, handles)
1708  % hObject    handle to displayFrontMicPreAmbientFile (see GCBO)
1709  % eventdata  reserved - to be defined in a future version of MATLAB
1710  % handles    structure with handles and user data (see GUIDATA)
1711
1712  function controlFrontMicPreAmbientChannel_Callback(hObject, eventdata, handles)
1713  % hObject    handle to controlFrontMicPreAmbientChannel (see GCBO)
1714  % eventdata  reserved - to be defined in a future version of MATLAB
1715  % handles    structure with handles and user data (see GUIDATA)
1716
1717  function displayFrontMicPostAmbientFile_Callback(hObject, eventdata, handles)
1718  % hObject    handle to displayFrontMicPostAmbientFile (see GCBO)
1719  % eventdata  reserved - to be defined in a future version of MATLAB
1720  % handles    structure with handles and user data (see GUIDATA)
1721
1722  function controlFrontMicPostAmbientChannel_Callback(hObject, eventdata, handles)
1723  % hObject    handle to controlFrontMicPostAmbientChannel (see GCBO)
1724  % eventdata  reserved - to be defined in a future version of MATLAB
1725  % handles    structure with handles and user data (see GUIDATA)
1726
1727  function displayTriggerFile_Callback(hObject, eventdata, handles)
1728  % hObject    handle to displayTriggerFile (see GCBO)
1729  % eventdata  reserved - to be defined in a future version of MATLAB
1730  % handles    structure with handles and user data (see GUIDATA)
1731
1732  % --- Executes on selection change in controlTriggerChannel.
```

```
function controlTriggerChannel_Callback(hObject, eventdata, handles) % Channels are now read-only in the Data Review Section
if isfield( handles, 'hPlotTrigger' )
    load( [ handles.signalPathName handles.signalFileName ] )
    handles.triggerChannelNumber = get( handles.controlTriggerChannel, 'Value' );

    if handles.useTrigger
        triggerEval = sprintf( 'handles.triggerSignal_Amp = Channel_%1.0f_Data;', handles.triggerChannelNumber );
        eval( triggerEval )
    else
        handles.triggerSignal_Amp =  0 * handles.signal_1;
        handles.triggerSignal_Amp( end ) = 1;
    end

    handles.triggerSignal_Amp = handles.triggerSignal_Amp( handles.idxTrimStart : handles.idxTrimStop );

    handles = plotData( handles );

    % Update handles structure
    guidata( hObject, handles );

end

function editTrigger_Callback(hObject, eventdata, handles)
% hObject    handle to editTrigger (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in checkboxTrigger.
function checkboxTrigger_Callback(hObject, eventdata, handles) % This checkbox is always checked and not exposed to the user

handles.useTrigger = get( handles.checkboxTrigger, 'Value');

if handles.useTrigger
    set( handles.controlTriggerChannel, 'Enable', 'on' )
else
    set( handles.controlTriggerChannel, 'Enable', 'off' )

    if isfield( handles, 'trigger' )
        handles.triggerSignal_Amp = 0 * handles.triggerSignal_Amp;
        handles.triggerSignal_Amp(end) = 1;
    end
end

handles = plotData( handles );

% Update handles structure
guidata( hObject, handles );

```

```
function controlPlayDriverSignal_Callback(hObject, eventdata, handles)
% hObject    handle to controlPlayDriverSignal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

sound( handles.signal_1, handles.signalFs )

% --- Executes on button press in controlPlayPassengerSignal.
function controlPlayPassengerSignal_Callback(hObject, eventdata, handles)
% hObject    handle to controlPlayPassengerSignal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

sound( handles.signal_2, handles.signalFs )

% --- Executes on button press in controlPlayAmbientimeTriggerSignal.
function controlPlayAmbientimeTriggerSignal_Callback(hObject, eventdata, handles)
% hObject    handle to controlPlayAmbientimeTriggerSignal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

sound( handles.ambient, handles.ambientFs )

function controlPlayFrontMicSignal_Callback(hObject, eventdata, handles)
% hObject    handle to controlPlayFrontMicSignal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over pushbuttonCompute.
function pushbuttonCompute_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonCompute (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function displayOutputFilePath_Callback(hObject, eventdata, handles)
% hObject    handle to displayOutputFilePath (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function displayCompliance_Callback(hObject, eventdata, handles)
% hObject    handle to displayCompliance (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbuttonExit.
function pushbuttonExit_Callback(hObject, eventdata, handles)

if ~ isempty( getappdata( 0, 'data' ) )
    rmappdata( 0, 'data' );
```

```
end

if ~ isempty( getappdata( 0, 'ink' ) )
    rmappdata( 0, 'ink' );
end

close
```

```
function [ idxCombos2Bands ] = idxCombinationsToBands( idxCombinations )

% This function take the band indices and converts to bands for readibility
% in reporting

idxCombinations (idxCombinations == 1) = 315 ;
idxCombinations (idxCombinations == 2) = 400 ;
idxCombinations (idxCombinations == 3) = 500 ;
idxCombinations (idxCombinations == 4) = 630 ;
idxCombinations (idxCombinations == 5) = 800 ;
idxCombinations (idxCombinations == 6) = 1000 ;
idxCombinations (idxCombinations == 7) = 1250 ;
idxCombinations (idxCombinations == 8) = 1600 ;
idxCombinations (idxCombinations == 9) = 2000 ;
idxCombinations (idxCombinations == 10) = 2500 ;
idxCombinations (idxCombinations == 11) = 3150 ;
idxCombinations (idxCombinations == 12) = 4000 ;
idxCombinations (idxCombinations == 13) = 5000 ;

idxCombos2Bands = idxCombinations ;

end
```

```
function [ handles ] = main_ComplianceInputPrep ( handles )
% This function allows for different input methods for testing purposes.
% Old testing methods have been removed, but the structure is still in
% place for future testing if needed.

%*************************************************************************
% SWITCHES (used in PreProcessing and Processing Code)
inputMethod = 'useGUIInput' ; %useGUIInput, useSpreadsheetInput, useCommandLinePrompt
report = 'on' ; % turn 'on' or 'off' to write intermediate results to xlsx files

% Switch for different input methods
switch inputMethod
    case 'useGUIInput'
        inputType = 1 ;
end

% If 'on', write intermediate results to xlsx files
switch report
    case 'on'
        disp( 'Intermediate results will be written to xlsx' )
        reporting = 1 ;
    case 'off'
        disp( 'Reporting of intermediate results is off' )
        reporting = 0 ;
end


%% DATA INPUT

% DATA INPUT FROM GUI
if inputType == 1;
    handles.data = getappdata( 0, 'data' );
    handles = ComplianceAnalysis_GUIPreprocessing( handles );

    drawnow
    if handles.stopButtonFlag
        % For stop button
        drawnow
        return;
    end

    if handles.outputFileCancel
        return
    end

end

%% Processing

if inputType == 1;
```

```
51        handles =  ComplianceAnalysis_GUIProcessing( handles );
52        drawnow
53        if handles.stopButtonFlag
54            % For stop button
55            drawnow
56            return;
57        end
58    end
59
60    end
61
62
63
64
65
```

```
function data = Max_SLM_A_Fast_44p1_kHz( pZ )

% pZ = Unweighted sound pressure, pa
% fs = Sampling frequency
% Frequency Weighting = Z
% Time Weighting = fast

fs = 44100; % Function is only valid for 44.1 kHz

%--------------------------------------------------------------------------
%  A-Weighting - Overall
%--------------------------------------------------------------------------

load AFilter_fs_44p1_kHz.mat B A % Load filters for A-weighting
pA = Compute_A_Weighting_44p1_kHz( pZ, B, A );

%--------------------------------------------------------------------------
%  Exponential Time Window Averaging - Overall
%--------------------------------------------------------------------------
% Time weighting must be done last since it operates on the pressure
% squared, not the pressure, hince a non-linear operation

spl = Compute_Fast_Time_Weighting_44p1_kHz( pA );

%--------------------------------------------------------------------------
% Statistical Metrics - Overall Levels
%--------------------------------------------------------------------------
endBuffer = round( 0.1 * fs ) ;

[ maxSPL, idxMaxSPL ] = max( spl( 1 : end - endBuffer ) );
% The "endBuffer" is to throw out the last few samples as
% these may have transient effects and additional 0.1*fs was added to the
% original signal to account for this in Compute_SPL_by_Vehicle_Side_forGUI

%--------------------------------------------------------------------------
%  Generate One-Third Octave Band Sub-Channels
%--------------------------------------------------------------------------

% Load one-third octave band filters from 25 Hz to 16 kHz
load FOBFiltersNHTSA Hd F0 % Note we are using this version to get 25 Hz to 16 kHz
% Each row contains the time signal associated with a specific 1/3 OB
idxStart = max( [ 1 ( idxMaxSPL - 5 * fs ) ] ); %Removing first bit of signal to decrease computation time
oneThirdOB_pA = Compute_One_Third_Octave_Bands_44p1_kHz_Hd( Hd, pA( idxStart : idxMaxSPL + endBuffer ) ); % Computing
past idxMaxSPL by endBuffer to avoid transient effects

%--------------------------------------------------------------------------
%  Exponential Time Window Averaging - One-Third Octave Bands
%--------------------------------------------------------------------------
% Time weighting must be done last since it operates on the pressure
% squared, not the pressure
```

```
50
51  % Initialize OBL
52  oneThirdOBSPLs = zeros( size( oneThirdOB_pA ) );
53  % Each row is a 1/3 OB, each column is the Level at the associated time
54
55  for ink = 1 : length( Hd )
56      oneThirdOBSPLs( ink, : ) = ...
57          Compute_Fast_Time_Weighting_44p1_kHz( oneThirdOB_pA( ink, : ) );
58  end
59
60  %--------------------------------------------------------------------------
61  % Statistical Metrics - One-Third Octave Bands
62  %--------------------------------------------------------------------------
63
64  % Statistics correlating to Overall Level Statistics, e.g. max of LAfMax
65  % At max relative to overall level
66  OTOBsAtMax = oneThirdOBSPLs( :, end - endBuffer); % This is correct, end - endBuffer  = idxMaxSPL for original signal
67
68  %--------------------------------------------------------------------------
69  % Format Results
70  %--------------------------------------------------------------------------
71
72  data.timeWeighting = 'fast';
73  data.frequencyWeighting = 'A';
74  data.fs = fs;
75  data.fc = F0;
76
77  % All Data Below are A-weighted
78  data.maxSPL = round( 10 * maxSPL ) / 10;
79  data.minSPL = [];
80  data.oneThirdOctaveBandLevelsAtMax = round ( 10 * OTOBsAtMax ) / 10;
81  data.oneThirdOctaveBandLevelsAtMin = [];
82
83
84
85
86
87
88
89
90
91
92
93
```

```
function data = Min_SLM_A_Fast_44p1_kHz( pZ )

% pZ = Unweighted sound pressure, pa
% fs = Sampling frequency
% Frequency Weighting = Z
% Time Weighting = fast

fs = 44100; % Function is only valid for 44.1 kHz

%--------------------------------------------------------------------------
%  A-Weighting - Overall
%--------------------------------------------------------------------------

load AFilter_fs_44p1_kHz.mat B A % Load filters for A-weighting
pA = Compute_A_Weighting_44p1_kHz( pZ, B, A );

%--------------------------------------------------------------------------
%  Exponential Time Window Averaging - Overall
%--------------------------------------------------------------------------
% Time weighting must be done last since it operates on the pressure
% squared, not the pressure, hince a non-linear operation

spl = Compute_Fast_Time_Weighting_44p1_kHz( pA );

%--------------------------------------------------------------------------
% Statistical Metrics - Overall Levels
%--------------------------------------------------------------------------

% [ maxSPL, idxMaxSPL ] = max( spl( 1 : end - round( 0.1 * fs ) ) );
% The "end - round( 0.1 * fs )" is to throw out the last few samples as
% these may have transient effects

% Need to offset to account for rise time, still a bit ambiguous
offset = round( fs * 1 ); % Offset by 1 Second
[ minSPL, idxMinSPL ] = min( spl( offset : end - round( 0.1 * fs ) ) );
idxMinSPL = idxMinSPL + offset - 1; % Correct for offset in indexing

%--------------------------------------------------------------------------
%  Generate One-Third Octave Band Sub-Channels
%--------------------------------------------------------------------------

% Load one-third octave band filters from 25 Hz to 16 kHz
load FOBFiltersNHTSA Hd F0 % Note we are using this version to get 25 Hz to 16 kHz
% Each row contains the time signal associated with a specific 1/3 OB
idxStart = max( [ 1 ( idxMinSPL - 5 * fs ) ] );
oneThirdOB_pA = Compute_One_Third_Octave_Bands_44p1_kHz_Hd( Hd, pA( idxStart : idxMinSPL ) );

%--------------------------------------------------------------------------
%  Exponential Time Window Averaging - One-Third Octave Bands
%--------------------------------------------------------------------------
```

```
% Time weighting must be done last since it operates on the pressure
% squared, not the pressure

% Initialize OBL
oneThirdOBSPLs = zeros( size( oneThirdOB_pA ) );
% Each row is a 1/3 OB, each column is the Level at the associated time

for ink = 1 : length( Hd )
    oneThirdOBSPLs( ink, : ) = ...
        Compute_Fast_Time_Weighting_44p1_kHz( oneThirdOB_pA( ink, : ) );
end

%--------------------------------------------------------------------------
% Statistical Metrics - One-Third Octave Bands
%--------------------------------------------------------------------------

% Statistics correlating to Overall Level Statistics, e.g. max of LAfMax
% OTOBsAtMax = oneThirdOBSPLs( :, idxMaxSPL );
OTOBsAtMin = oneThirdOBSPLs( :, end );

%--------------------------------------------------------------------------
% Format Results
%--------------------------------------------------------------------------

data.timeWeighting = 'fast';
data.frequencyWeighting = 'A';
data.fs = fs;
data.fc = F0;

% All Data Below are A-weighted
data.maxSPL = [];
data.minSPL = round( 10 * minSPL ) / 10 ;
data.oneThirdOctaveBandLevelsAtMax = [];
data.oneThirdOctaveBandLevelsAtMin = round( 10 * OTOBsAtMin ) / 10 ;
```

```
function [ passFail, ocPass  ] = passFailCheck2_Avg( signalPerformance )

dimension = 0;

for ink = 1 : 5
    dim = size( signalPerformance{ ink }, 2 );
    dimension = max( dim, dimension );
end

% Initialize matrix
signalPerformancePre = zeros( 5, dimension );

% Add performance from the samples for each operation condition
for ink = 1 : 5
    signalPerformancePre( ink,  : ) = signalPerformance{ ink } ; % Bringing performance check into double from cell
    array
end

ocPass = sum( signalPerformancePre, 2 );

% To pass back and evaluate for pass/fail (to pass, it must equal 5)
passFail = length( ocPass( ocPass > 0 ) );


```

```
clc;

%Ruling requirement data

fc = [ 315 400 500 630 800 1000 1250 1600 2000 2500 3150 4000 5000 ];

minReq4Band = cell( 1, 5 );
minReq4Band{ 1 } = [ 42 41 43 43 44 44 45 41 42 40 37 35 33 ];
minReq4Band{ 2 } = [ 39 39 40 40 41 41 42 39 39 37 34 32 31 ];
minReq4Band{ 3 } = [ 45 44 46 46 47 47 48 44 45 43 40 38 36 ];
minReq4Band{ 4 } = [ 52 51 52 53 53 54 54 51 51 50 47 45 43 ];
minReq4Band{ 5 } = [ 56 55 57 57 58 58 59 55 55 54 51 49 47 ];

minReq2Band = cell( 1, 5 );
minReq2Band{ 1 } = [ 39 48 ];
minReq2Band{ 2 } = [ 40 44 ];
minReq2Band{ 3 } = [ 42 51 ];
minReq2Band{ 4 } = [ 47 57 ];
minReq2Band{ 5 } = [ 52 62 ];

save rulingRequirementData.mat


```

```
function handles = selectBatchDirectory( handles )

% User selects the directory with files to batch import
handles.batchDirectory = uigetdir ;

% If canceled out of directory selection window
try
batchFileDir = dir ( handles.batchDirectory ) ;
catch
    handles.batchFileCancel = true ;
    return
end

% Folder names
folderStationary = '\Stationary\' ;
folder10kmh = '\10kmh\' ;
folder20kmh = '\20kmh\' ;
folder30kmh = '\30kmh\' ;
folderReverse = '\Reverse\' ;

% Pull files from folders
try

    allFilesInStationaryBatch = dir ( strcat( handles.batchDirectory, folderStationary ) ) ;
    filesInStationaryBatch = allFilesInStationaryBatch( ~ismember( { allFilesInStationaryBatch.name }, { '.','..' } )
    );
    handles.pathStationary = strcat( handles.batchDirectory, folderStationary ) ;

    allFilesIn10kmhBatch = dir ( strcat( handles.batchDirectory, folder10kmh ) ) ;
    filesIn10kmhBatch = allFilesIn10kmhBatch( ~ismember( { allFilesIn10kmhBatch.name }, { '.','..' } ) );
    handles.path10kmh = strcat( handles.batchDirectory, folder10kmh ) ;

    allFilesIn20kmhBatch = dir ( strcat( handles.batchDirectory, folder20kmh ) ) ;
    filesIn20kmhBatch = allFilesIn20kmhBatch( ~ismember( { allFilesIn20kmhBatch.name }, { '.','..' } ) );
    handles.path20kmh = strcat( handles.batchDirectory, folder20kmh ) ;

    allFilesIn30kmhBatch = dir ( strcat( handles.batchDirectory, folder30kmh ) ) ;
    filesIn30kmhBatch = allFilesIn30kmhBatch( ~ismember( { allFilesIn30kmhBatch.name }, { '.','..' } ) );
    handles.path30kmh = strcat( handles.batchDirectory, folder30kmh ) ;

    allFilesInReverseBatch = dir ( strcat( handles.batchDirectory, folderReverse ) ) ;
    filesInReverseBatch = allFilesInReverseBatch( ~ismember( { allFilesInReverseBatch.name }, { '.','..' } ) );
    handles.pathReverse = strcat( handles.batchDirectory, folderReverse ) ;

    handles.batchFileNames = [filesInStationaryBatch; filesIn10kmhBatch; filesIn20kmhBatch; filesIn30kmhBatch;
    filesInReverseBatch ] ;

catch
    handles.folderStructureError = 1 ;
    return
```

```
49    end
50
51
52    end
```

```
function [ signalPerformance, bandSumTracking ] = signalPerformance2_avg( threshold_dBA, idxCombinations, signal_dBA  )
% This function compares the signal to the 2-band requirements

signalPerformance = zeros( 1, size( idxCombinations, 1 ) );

% Looping through all combinations of 2 bands to see if any pass the 2-band threshold requirement
for ink = 1 : size( idxCombinations, 1 )

    idx = idxCombinations( ink, idxCombinations( ink, : ) > 0 );% Of the combinations checking the "ink" row to make sure all indices are greater than 0.
                                                                % They should always be greater than 0!Check may not be needed
    for lead = 1 : size( signal_dBA, 1 )

        if length( threshold_dBA ) > size( idxCombinations, 2 ) % If the size of the threshold (requirement) is larger than the size of the combinations (again, should always be true).
            thresholdExceedance = signal_dBA( lead, idx ) - threshold_dBA( idx( 1 ) ) ; % The threshold is the minimum SPL levels for the 2-band requirement
        else
            thresholdExceedance = signal_dBA( lead, idx ) - threshold_dBA( 1 ) ;
        end

        % Bands Spanned, Number of Bands, Number of Bands Detectable, Band Sum
        numBandsDetected = length( thresholdExceedance( thresholdExceedance >= 0 ) ); % Since signal-threshold, looking for values that are greater than or equal to 0
        if numBandsDetected == 2 %2 bands
            bandSum = 10 * log10( 10^( signal_dBA( lead, idx( 1 ) )/10 ) + 10^( signal_dBA( lead, idx( 2 ) )/ 10 ) );
            bandSum = round( 10 * bandSum ) / 10 ; % Round to 1 decimal point

            if bandSum >= threshold_dBA( 2 ) % Comparing the band sum against the band sum requirement
                signalPerformance( lead, ink ) = 1;
                bandSumTracking( lead, ink ) = bandSum ;
            else
                signalPerformance( lead, ink ) = 0;
                bandSumTracking( lead, ink ) = bandSum ;
            end

        else
            signalPerformance( lead, ink ) = 0;
            bandSumTracking( lead, ink ) = 0 ;
        end
    end
end
```

```
function [ signalPerformance ] = signalPerformance4_avg( threshold_dBA, idxCombinations, signal_dBA  )
% This function compares the signal to the requirement

% Initializing signalPerformance variable
signalPerformance = zeros( 1, size( idxCombinations, 1 ) );

%looping through all combinations of 4 bands to see if any pass the 4-band threshold requirement
for ink = 1 : size( idxCombinations, 1 )

    idx = idxCombinations( ink, idxCombinations( ink, : ) > 0 ); % Of the combinations checking the "ink" row to
    make sure all indices are greater than 0.
                                                                 % They should always be greater than 0!Check may
                                                                 not be needed

    for lead = 1 : size( signal_dBA, 1 )

        if length( threshold_dBA ) > size( idxCombinations, 2 ) % If the size of the threshold (requirement) is
        larger than the size of the combinations (again, should always be
        true).
            thresholdExceedance = signal_dBA( lead, idx ) - threshold_dBA( idx ) ; % The threshold is the minimum
            SPL levels for the 4-band requirement
        else
            thresholdExceedance = signal_dBA( lead, idx ) - threshold_dBA ;
        end


        % Bands Spanned, Number of Bands, Number of Bands Detectable
        numBandsDetected = length( thresholdExceedance( thresholdExceedance >= 0 ) ); % Since signal-threshold,
        looking for values that are greater than or equal to 0
        if numBandsDetected == 4 %4 bands
            signalPerformance( lead, ink ) = 1;
        else
            signalPerformance( lead, ink ) = 0;
        end

    end

end
```

```
% Validate SLM Calibration
% Use this to confirm SLM is working correctly

clear
clc

fs = 44100; % Hz
t = 0: 1/fs : 10;  % Time for series, seconds
f = 1000; % Hz

y = sin( 2 * pi * f * t ); % Creating a test signal
yrms = sqrt( mean( y.^2 ) ); % RMS value of time signal

spl_have = 20 * log10( yrms / 20e-6 );% Computed theoretical SPL for the signal
spl_want = 60;% Target SPL

c = 10^((  spl_want - spl_have ) / 20 ); % Calibration coefficient
y = c * y; % Applying calibration coefficient to scale test signal to desired level


% Checking accuracy of SLM in code
results = Max_SLM_A_Fast_44p1_kHz( y ) ;

spl_out = results.maxSPL
```

```
function [ volumeShift, pass, handles ] = volumeShift( dataSample, minReq0kph, handles )

%Initialize cells
meanDataSample = cell( 1, 5 );
normalizedDataSample = cell( 1, 5 );
bandSum = zeros( 1, 5 );
shift = zeros( 1, 5 );
pass = zeros( 1, 5 );

% Normalize levels according to S7.6.2
% For each critical operating scenario, normalize the levels of the 13
% one-third octave bands by subtracting the corresponding minimum SPL
% values specified in Table 1 for the stationary operating condition from
% each of the one-third octave band averages calculated in S7.6.1.

%Op Conditions
% 1 - Reverse
% 2 - Stationary
% 3 - 10 km/h
% 4 - 20 km/h
% 5 - 30 km/hfor ink = 1 : 5  % looping through op types

for ink = 1 : 5
    meanDataSample{ ink }= round( 10 * mean( dataSample{ ink }, 1 ) ) / 10;
    normalizedDataSample{ ink } = meanDataSample{ ink } - minReq0kph;
    inBetween = 10 .^ ( normalizedDataSample{ ink } ./ 10 );
    inBetween = round( 10 * inBetween ) / 10; % Round to 1 decimal point
    bandSum( ink ) = 10 * log10( sum( inBetween, 2 ) );
    bandSum = round( 10 * bandSum ) / 10; % Round to 1 decimal point
end

for ink = 2 : 4
    shift( ink ) = bandSum( ink + 1 ) - bandSum( ink );
    if  shift( ink ) >= 3
        pass( ink ) = 1;
    end
end

if sum( pass ) == 3
    volumeShift = 1;
else
    volumeShift = 0;
end

handles.meanDataSample = meanDataSample ;
handles.RelVolMinReq = minReq0kph ;
handles.normalizedDataSample = normalizedDataSample ;
handles.normalizedBandSum = bandSum ;
handles.shift = shift ;
handles.pass = pass ;
```