```
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/h for 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( fileNumbersPassby ) ] );
    end

    numberInc = numberInc + 1 ;

else

    numberInc = 1 ;

end

handles.numberInc = numberInc ;
```

```
98   %% Data Prep
99
100  for ink = 1 : 5
101
102      % ----------------------------------------------------------------
103      % For stop button
104      drawnow
105      if get(handles.pushbuttonStop, 'userdata') % stop condition
106          handles.stopButtonFlag = true ;
107          return;
108      end
109      % ----------------------------------------------------------------
110
111      handles.opCondForLoop = opCondOptions{ ink } ; % Cell Array {'Reverse'}    {'Stationary'}    {'10 km/h'}    {'20
         km/h'}    {'30 km/h'}
112
113      %% Select signal data and process through SLM
114
115      %-----------------------------------------------------------------------------------------
116      %Final Rule: S7.3.3(a)
117      %For each valid test sound file identified in S7.3.2, determine a maximum overall
118      %SPL value, in decibels. The SPL value will be reported to the nearest tenth of a decibel.
119      %-----------------------------------------------------------------------------------------
120
121      tic
122      handles = Compute_SPL_by_Vehicle_Side_forGUI( handles ); % Each call to this handles a single operating condition
         (see ~ line 111 above)
123      drawnow
124      if handles.stopButtonFlag
125          % For stop button
126          return;
127      end
128
129      %*********************************************************************
130      %*********************************************************************
131      % handles conversion to variables
132      resultsDriver = handles.driverSPL ;
133      resultsPassenger = handles.passengerSPL ;
134      resultsFront = handles.frontSPL ;
135      %*********************************************************************
136      %*********************************************************************
137
138      handles.opCondSimple = opCondOptionsSimple{ ink } ;
139
140      eval( [ sprintf( 'resultsDriver_%s', handles.opCondSimple ) ' = resultsDriver;' ] )
141      eval( [ sprintf( 'resultsPassenger_%s', handles.opCondSimple ) ' = resultsPassenger;' ] )
142      eval( [ sprintf( 'resultsFront_%s', handles.opCondSimple ) ' = resultsFront;' ] )
143
144      driverInputFileName = resultsDriver.inputFileNames ;
145      driverAmbientFileName = resultsDriver.ambientFileNames ;
```

```
    driverAmbientSide = resultsDriver.ambientLowerSide ;
    passengerInputFileName = resultsPassenger.inputFileNames ;
    passengerAmbientFileName = resultsPassenger.ambientFileNames ;
    passengerAmbientSide = resultsPassenger.ambientLowerSide ;

    if ink == 2 %for stationary only

        frontInputFile = resultsFront.inputFileNames ;
        frontAmbientInputFile = resultsFront.ambientFileNames ;
        fileNames.side{ ink } = resultsFront.inputFileNames ;
        fileNames.sideambient{ ink } = resultsFront.ambientFileNames ;

    end

%     try
%         save( 'timeSeriesResults.mat', sprintf( 'resultsDriver_%s', handles.opCondSimple ), sprintf( 'resultsPassenger_%s', handles.opCondSimple ), sprintf( 'resultsFront_%s', handles.opCondSimple ), '-append' );
%     catch
%         save( 'timeSeriesResults.mat', sprintf( 'resultsDriver_%s', handles.opCondSimple ), sprintf( 'resultsPassenger_%s', handles.opCondSimple ), sprintf( 'resultsFront_%s', handles.opCondSimple ) );
%         toc
%
%     end

    %% Choose four samples within 2 dB of each other:
    % ----------------------------------------------------------------
    % For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
    % ----------------------------------------------------------------

    %-----------------------------------------------------------------------------------------------
    %Final Rule: S7.3.3(b)
    % Compare the first four left side maximum overall SPL values. Of the four SPL values
    % calculate the difference between the largest and smallest maximum SPL values. The same
    % process will be used to determine the difference between the largest and smallest maximum
    % SPL values for the first four right side maximum SPL values. If the difference values on
    % the left and right sides of the test vehicle are both less than or equal to 2.0 dB, then
    % the eight sound files associated with the first four valid tests will be used for the final
    % one-third octave band evaluation in accordance with S7.3.4. and S7.3.5. If the first four
    % test sound files on each side 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 both
    % microphone locations are within 2 dBA for the same sequence number recordings for both locations.
    %-----------------------------------------------------------------------------------------------
    disp( 'Choosing 4 within 2 dB.' )

    set( handles.textProcessingStep, 'String', 'Choosing 4 samples within 2 dB' );
```

```matlab
    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 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 SPL Results - 4 Samples'} ;
        overallHeaderDriverVehicle = {'Driver Side Vehicle Results (dBA)'} ;
```

```
243         overallHeaderDriverAmbient = {'Driver Side Ambient Results (dBA)'} ;
244         overallHeaderPassengerVehicle = {'Passenger Side Vehicle Results (dBA)'} ;
245         overallHeaderPassengerAmbient = {'Passenger Side Ambient Results (dBA)'} ;
246         FOBHeader = {'FOB Results - 4 Samples'} ;
247         FOBHeaderDriverVehicle = {'Driver Side Vehicle Results (dBA)'} ;
248         FOBHeaderDriverAmbient = {'Driver Side Ambient Results (dBA)'} ;
249         FOBHeaderPassengerVehicle = {'Passenger Side Vehicle Results (dBA)'} ;
250         FOBHeaderPassengerAmbient = {'Passenger Side Ambient Results (dBA)'} ;
251
252         header_overallFront = {'Overall SPL Results - 4 Samples, FRONT'} ;
253         header_overallFrontVehicle = {'Front Side Vehicle Results (dBA)'} ;
254         header_overallFrontAmbient = {'Front Side Ambient Results (dBA)'} ;
255         header_frontFOB = {'FOB Results - 4 Samples, FRONT'} ;
256         header_FOBFrontVehicle = {'Front Side Vehicle Results (dBA)'} ;
257         header_FOBFrontAmbient = {'Front Side Ambient Results (dBA)'} ;
258
259         % Results
260         xlswrite (fileNamePath,overallHeader, '4Samples', 'A3')
261         xlswrite (fileNamePath,overallHeaderDriverVehicle, '4Samples', 'A4')
262         xlswrite(fileNamePath, resultsDriver.vehicleOverallLevels_dBA, '4Samples', 'A5')
263         xlswrite (fileNamePath,overallHeaderDriverAmbient, '4Samples', 'A6')
264         xlswrite(fileNamePath, resultsDriver.ambientOverallLevels_dBA, '4Samples', 'A7')
265         xlswrite (fileNamePath,overallHeaderPassengerVehicle, '4Samples', 'A8')
266         xlswrite(fileNamePath, resultsPassenger.vehicleOverallLevels_dBA, '4Samples', 'A9')
267         xlswrite (fileNamePath,overallHeaderPassengerAmbient, '4Samples', 'A10')
268         xlswrite(fileNamePath, resultsPassenger.ambientOverallLevels_dBA, '4Samples', 'A11')
269
270         xlswrite (fileNamePath, FOBHeaderDriverVehicle, '4Samples', 'A14')
271         xlswrite (fileNamePath, oBs, '4Samples', 'A15')
272         xlswrite (fileNamePath, resultsDriver.vehicleFOBLevels_dBA, '4Samples', 'A16')
273         xlswrite (fileNamePath, FOBHeader, '4Samples', 'A13')
274
275         xlswrite (fileNamePath, FOBHeaderDriverAmbient, '4Samples', 'A21')
276         xlswrite (fileNamePath, oBs, '4Samples', 'A22')
277         xlswrite (fileNamePath, resultsDriver.ambientFOBLevels_dBA, '4Samples', 'A23')
278
279         xlswrite (fileNamePath, FOBHeaderPassengerVehicle, '4Samples', 'A28')
280         xlswrite (fileNamePath, oBs, '4Samples', 'A29')
281         xlswrite (fileNamePath, resultsPassenger.vehicleFOBLevels_dBA, '4Samples', 'A30')
282
283         xlswrite (fileNamePath, FOBHeaderPassengerAmbient, '4Samples', 'A35')
284         xlswrite (fileNamePath, oBs, '4Samples', 'A36')
285         xlswrite (fileNamePath, resultsPassenger.ambientFOBLevels_dBA, '4Samples', 'A37')
286
287         xlswrite (fileNamePath,sheetSamplesHeader, '4Samples', 'A1')
288
289         % Add front samples for stationary
290          if strcmpi( handles.opCondSimple, 'Stationary' ) % Only do front if Stationary and front was computed
291             % headers
292              xlswrite (fileNamePath, header_overallFront, '4Samples', 'O3')
```

```
             xlswrite (fileNamePath, header_overallFrontVehicle, '4Samples', 'O4')
             xlswrite (fileNamePath, header_overallFrontAmbient, '4Samples', 'O6')
             xlswrite (fileNamePath, header_frontFOB, '4Samples', 'O13')
             xlswrite (fileNamePath, header_FOBFrontVehicle, '4Samples', 'O14')
             xlswrite (fileNamePath, oBs, '4Samples', 'O15')
             xlswrite (fileNamePath, header_FOBFrontAmbient, '4Samples', 'O21')
             xlswrite (fileNamePath, oBs, '4Samples', 'O22')

             % data
             xlswrite (fileNamePath, resultsFront.vehicleOverallLevels_dBA, '4Samples', 'O5')
             xlswrite (fileNamePath, resultsFront.ambientOverallLevels_dBA, '4Samples', 'O7')
             xlswrite (fileNamePath, resultsFront.vehicleFOBLevels_dBA, '4Samples', 'O16')
             xlswrite (fileNamePath, resultsFront.ambientFOBLevels_dBA, '4Samples', 'O23')

         end

    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 SPL Results Corrected for Ambient - 4 Samples'} ;
        ambiCorOverallHeaderDriverVehicle = {'Driver Side Vehicle Results - Corrected for Ambient (dBA)'} ;
        ambiCorOverallHeaderDriverAmbient = {'Driver Side Ambient Results (dBA)'} ;
        ambiCorOverallHeaderPassengerVehicle = {'Passenger Side Vehicle Results - Corrected for Ambient (dBA)'} ;
        ambiCorOverallHeaderPassengerAmbient = {'Passenger Side Ambient Results (dBA)'} ;

        ambiCorFOBHeader = {'FOB SPL Results Corrected for Ambient - 4 Samples'} ;
        ambiCorFOBHeaderDriverVehicle = {'Driver Side Vehicle Results - Corrected for Ambient (dBA)'} ;
        ambiCorFOBHeaderDriverAmbient = {'Driver Side Ambient Results (dBA)'} ;
        ambiCorFOBHeaderPassengerVehicle = {'Passenger Side Vehicle Results - Corrected for Ambient (dBA)'} ;
        ambiCorFOBHeaderPassengerAmbient = {'Passenger Side Ambient Results (dBA)'} ;

        ambiCorHeader_overallFront = {'Overall SPL Results Corrected for Ambient - 4 Samples, FRONT'} ;
        ambiCorHeader_overallFrontVehicle = {'Front Side Vehicle Results - Corrected for Ambient (dBA)'} ;
        ambiCorHeader_overallFrontAmbient = {'Front Side Ambient Results (dBA)'} ;
        ambiCorHeader_frontFOB = {'FOB SPL Results - 4 Samples, FRONT'} ;
        ambiCorHeader_FOBFrontVehicle = {'Front Side Vehicle Results - Corrected for Ambient (dBA)'} ;
        ambiCorHeader_FOBFrontAmbient = {'Front Side Ambient Results (dBA)'} ;

        % 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')
        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')

        % Add front samples for stationary
         if strcmpi( handles.opCondSimple, 'Stationary' ) % Only do front if Stationary and front was computed
            % headers
            xlswrite (fileNamePath, ambiCorHeader_overallFront, '4SamplesCorrected', 'O3')
            xlswrite (fileNamePath, ambiCorHeader_overallFrontVehicle, '4SamplesCorrected', 'O4')
            xlswrite (fileNamePath, ambiCorHeader_overallFrontAmbient, '4SamplesCorrected', 'O6')
            xlswrite (fileNamePath, ambiCorHeader_frontFOB, '4SamplesCorrected', 'O13')
            xlswrite (fileNamePath, ambiCorHeader_FOBFrontVehicle, '4SamplesCorrected', 'O14')
            xlswrite (fileNamePath, oBs, '4SamplesCorrected', 'O15')
            xlswrite (fileNamePath, ambiCorHeader_FOBFrontAmbient, '4SamplesCorrected', 'O21')
            xlswrite (fileNamePath, oBs, '4SamplesCorrected', 'O22')

            % data
            xlswrite (fileNamePath, resultsFrontCorrected.vehicleOverallLevels_dBA, '4SamplesCorrected', 'O5')
            xlswrite (fileNamePath, resultsFrontCorrected.ambientOverallLevels_dBA, '4SamplesCorrected', 'O7')
            xlswrite (fileNamePath, resultsFrontCorrected.vehicleFOBLevels_dBA, '4SamplesCorrected', 'O16')
            xlswrite (fileNamePath, resultsFrontCorrected.ambientFOBLevels_dBA, '4SamplesCorrected', 'O23')
         end


        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 SPL Results - Corrected for Ambient, Averaged' } ;
        AvgOverallHeaderDriverVehicle = {'Driver Side Vehicle Results - Corrected for Ambient, Averaged (dBA)'} ;
        AvgOverallHeaderPassengerVehicle = {'Passenger Side Vehicle - Corrected for Ambient, Averaged (dBA)'} ;
        AvgFOBHeader = {'FOB SPL Results - Corrected for Ambient, Averaged'} ;
        AvgFOBHeaderDriverVehicle = {'Driver Side Vehicle Results - Corrected for Ambient, Averaged (dBA)'} ;
        AvgFOBHeaderPassengerVehicle = {'Passenger Side Vehicle Results - Corrected for Ambient, Averaged (dBA)'} ;

        avg_OverallFront = {'Overall SPL Results - Corrected for Ambient, Averaged, FRONT'} ;
        avg_OverallFrontVehicle = {'Front Side Vehicle Results - Corrected for Ambient, Averaged (dBA)'} ;
        avg_FrontFOB = {'FOB SPL Results - Corrected for Ambient, Averaged, FRONT (dBA)'} ;
        avg_FOBFrontVehicle = {'Front Side Vehicle Results - Corrected for Ambient, Averaged (dBA)'} ;

        % 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')

         % Add front samples for stationary
         if strcmpi( handles.opCondSimple, 'Stationary' ) % Only do front if Stationary and front was computed
            % headers
            xlswrite (fileNamePath, avg_OverallFront, 'Averaged', 'O3')
            xlswrite (fileNamePath, avg_OverallFrontVehicle, 'Averaged', 'O4')
            xlswrite (fileNamePath, avg_FrontFOB, 'Averaged', 'O9')
            xlswrite (fileNamePath, avg_FOBFrontVehicle, 'Averaged', 'O10')
            xlswrite (fileNamePath, oBs, 'Averaged', 'O11')

            % data
            xlswrite (fileNamePath, frontSideOverallAvg_dBA , 'Averaged', 'O5')
            xlswrite (fileNamePath, frontSideThirdOctaveAvg_dBA , 'Averaged', 'O12')

         end

        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)
    %--------------------------------------------------------------------------------------------------

```

```matlab
        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 SPL Results Corrected for Ambient, Averaged, Lower Side (dBA)' } ;
            finalFOBHeader = {'Final FOB SPL Results Corrected for Ambient, Averaged, Lower Side (dBA)'} ;
            lowerSideHeader = {lowerSide} ;

            % Results
            xlswrite (fileNamePath,lowerSideHeader, 'FinalSample', 'A3')
            xlswrite (fileNamePath,finalOverallHeader, 'FinalSample', 'A4')
            xlswrite(fileNamePath, finalOverallLevels_dBA, 'FinalSample', 'A5')

            xlswrite (fileNamePath, finalFOBHeader, 'FinalSample', 'A7')
            xlswrite (fileNamePath, oBs, 'FinalSample', 'A8')
            xlswrite (fileNamePath, finalThirdOBlevels_dBA, 'FinalSample', 'A9')

            xlswrite (fileNamePath,sheetFinalOverallHeader, 'FinalSample', 'A1')

        end

        %Prepare to pass data to handles to be used in the later analyses

        %Frequencies up to this point are from 25-16000 Hz, need to pass 315-5000
        %Hz for 4 band testing (NHTSA), 315-3150 Hz for 2 band (alliance)
        NHTSAsampleData{ ink } = finalThirdOBlevels_dBA  ;
        allianceSampleData{ ink } = finalThirdOBlevels_dBA ( 1 : 11 ) ;
        NHTSAFront{ ink } = frontSideThirdOctaveAvg_dBA ;
```

```
    micChoice{ ink } = lowerSide ;

    if strcmp( lowerSide,  'Driver' )
        fileNames.side{ ink } = driverInputFileName ;
        fileNames.sideambient{ ink } = driverAmbientFileName ;
        fileNames.ambientSide{ ink } = resultsDriver.ambientLowerSide;
        fileNames.side4{ ink } = resultsDriver.inputFileNames( idx );
        fileNames.sideambient4{ ink } = resultsDriver.ambientFileNames( idx );
        fileNames.ambientSide4{ ink } = resultsDriver.ambientLowerSide( idx );
        fileNames.frontambientSide{ ink } = { [] };

    else
        fileNames.side{ ink } = passengerInputFileName ;
        fileNames.sideambient{ ink } = passengerAmbientFileName ;
        fileNames.ambientSide{ ink } = resultsPassenger.ambientLowerSide;
        fileNames.side4{ ink } = resultsPassenger.inputFileNames( idx );
        fileNames.sideambient4{ ink } = resultsPassenger.ambientFileNames( idx );
        fileNames.ambientSide4{ ink } = resultsPassenger.ambientLowerSide( idx );
        fileNames.frontambientSide{ ink } = { [] };

    end

    if ink == 2

        fileNames.front{ ink } = frontInputFile ;
        fileNames.frontambient{ ink } = frontAmbientInputFile ;
        fileNames.side { ink }= resultsFront.inputFileNames ;
        fileNames.sideambient { ink } = resultsFront.ambientFileNames ;
        fileNames.frontambientSide{ ink } = resultsFront.ambientLowerSide;
        fileNames.front4{ ink } = resultsFront.inputFileNames( idx );
        fileNames.frontambient4{ ink } = resultsFront.ambientFileNames( idx );
        fileNames.frontambientSide4{ ink } = resultsFront.ambientLowerSide( idx );

    end

end

handles.NHTSAsampleData = NHTSAsampleData;
handles.allianceSampleData = allianceSampleData;
handles.NHTSAFront = NHTSAFront;
handles.micChoice = micChoice;
handles.fileNames = fileNames;
handles.Make = Make;
handles.model = Model;
handles.year = year;
```