```
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' )
load( 'rulingRequirementData.mat', 'minReqBandMinValbyBand' )
```

```
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 4-Band 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
% Equals number of components + minimum numb spaces e.g. 3 components + 2 spaces = 5
% Not really needed now, but ok to leave - this code was used during validataion and beta testing

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

            if isempty( NHTSAsignalPerformance{ ink } )
                NHTSAsignalPerformance{ ink } = zeros( size( idxCombinations' ) );
            end
        end
    end
end

% Checking to see if the results from the signalPerfomance function (to
% compare the signal to the requirement) pass or fail
[ passFail, ocPass ] = passFailCheck2_Avg( NHTSAsignalPerformance );

%Report status
if passFail > 0
    if passFail == 5
        fprintf( 'The %s %s %s passed the 4-band test\n\n', year, make, model )
    else
        fprintf( 'The %s %s %s did not pass the 4-band test\n\n', year, make, model)
    end

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

    fprintf( '\n' );

else
   fprintf( 'The %s %s %s did not pass the 4-band test\n\n', year, make, model)
end

%% ALLIANCE 2 BAND TEST

% ----------------------------------------------------------------
% For stop button
    drawnow
    if get(handles.pushbuttonStop, 'userdata') % stop condition
        handles.stopButtonFlag = true ;
        return;
    end
% ----------------------------------------------------------------
%-------------------------------------------------------------------------------
%Final Rule S7.3.5:
% (e)For alerts designed to meet the two one-third octave band requirements.
%  (i)Select the two highest one-third octave bands that are non-adjacent
%  to - WILL "HIGHEST" WORDING BE CHANGED IN FINAL? -
%   each other and within the range of 315 Hz up to and including 3150Hz 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 paragraph (ii), below.
%  (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 2-Band 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.

%4 band - Final Rule S5.1.1.2
%2-band - Final Rule S5.2.1
% Table 1 for Stationary (less than 10 km/h): 39, 39, 40, 40, 41, 41, 42, 39, 39, 37, 34, 32, 31

centralMicPerformance = zeros( 1, size( idxCombinations, 1 ) );
centralMicPerformance2 = zeros( 1, size( idxCombinations2, 1 ) );

if exist( 'NHTSAFront', 'var' )
    centralMic  = NHTSAFront ;
end

if isempty( centralMic{ 2 } ) % This is hard coded to "2" becuase looking at idle condition
    fprintf( 'This vehicle does not pass the directivity test for either the 4-band or the 2-band test\n\n');
    directivity = { 'Fail'; 'Fail' };
    dPass = 0;
    dPass2 = 0;
    central4bandData = zeros (1, 13);
    central2bandData = zeros (1, 11);

else

    if any( any( centralMic{ 2 } ) ) == 0
        % First check if there are ANY nonzero values in the data (yes = 1),
        % then check if the answer is nonzero (yes = 1), then check if the answer == 0,
        % if no (0) then there are results, if yes (1) then there are no nonzero values
        fprintf( 'This vehicle does not pass the directivity test for either the 4-band or the 2-band test\n\n');
        directivity = { 'Fail'; 'Fail' };
        dPass = 0;
        dPass2 = 0;

    else

        central4bandData = centralMic{ 2 }(:, 1 : 13 );
        central2bandData = centralMic{ 2 } (:, 1: 11 );

        % 4-band test
        for chalk = 1 : size( central4bandData, 1 )
            centralMicPerformance( chalk, : ) = signalPerformance4_avg( minReq4Band{ 2 }, idxCombinations,
            central4bandData( chalk , : ) );
        end

        % 2-band test
        for chalk = 1 : size( central2bandData, 1 )
            %centralMicPerformance2( chalk, : ) = signalPerformance2_avg( minReq2Band{ 2 }, idxCombinations2,
            central2bandData( chalk, : ) );
```

```
            [ centralMicPerformance2( chalk, : ) bandSumTrackingDirectivity( chalk, : )]= signalPerformance2_avg( minReq2Band{ 2 }, idxCombinations2, central2bandData( chalk, : ) );
        end

        % Directivity test
        [ directivityPass, dPass, dPass2 ] = directivityTest_avg( centralMicPerformance, centralMicPerformance2 );

        if isempty( dPass )
            dPass = 0;
        end

        if isempty( dPass2 )
            dPass2 = 0;
        end

        if directivityPass == 0
            fprintf( 'This vehicle does not pass the directivity for either the 4-band or the 2-band test\n\n');
            directivity = { 'Fail'; 'Fail' };
            directivityStatus = {'This vehicle does not pass the directivity for either the 4-band or the 2-band test' };

        elseif directivityPass == 2
            fprintf( 'This vehicle passes the directivity test for the 4-band test but not for the 2-band test\n\n');
            directivity = { 'Pass'; 'Fail' };
            directivityStatus = { 'This vehicle passes the directivity test for the 4-band test but not for the 2-band test' };

        elseif directivityPass == 1
            fprintf( 'This vehicle passes the directivity test for the 2-band test but not for the 4-band test\n\n');
            directivity = { 'Fail'; 'Pass' };
            directivityStatus = {'This vehicle passes the directivity test for the 2-band test but not for the 4-band test' };

        else
            fprintf( 'This vehicle passes the directivity test for both the 4-band test and the 2-band test\n\n');
            directivity = { 'Pass'; 'Pass' };
            directivityStatus = {'This vehicle passes the directivity test for both the 4-band test and the 2-band test' };

        end

    end

end

% For logging
fourBandCombosLabel = {'4 Band Combinations' } ;
twoBandCombosLabel = {'2 Band Combinations' };
passFailStatusLabel = { 'Pass/Fail' } ;
```

```
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' } ;
        if exist( 'central4bandData', 'var')
stationarySignal = ( round( central4bandData * 10) / 10 ) ;
        else
            disp('ooops')
        end
directivityThreshold = ( round( minReq4Band{ 2 } * 10) / 10 ) ;
directivityDifference = ( round( ( central4bandData - minReq4Band{ 2 } ) * 10 ) / 10 );

idxCombos2Bands = idxCombinationsToBands( idxCombinations ) ;
idxCombos2Bands2 = idxCombinationsToBands( idxCombinations2 ) ;

fourBandCombos = idxCombos2Bands ;
twoBandCombos = idxCombos2Bands2 ;

passFailStatus4Band = centralMicPerformance' ;
passFailStatus2Band = centralMicPerformance2' ;


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

% Final Rule S7.6
volumeChangeSample = cell( 1, 5 );

for ink = 1 : 5
    if isempty( NHTSAsampleData{ ink } )
        volumeChangeSample{ ink } = [ 5 5 5 5 5 5 5 5 5 5 5 5 5 ];
    else
        volumeChangeSample{ ink } = NHTSAsampleData{ ink };
    end
end

% Relative Volume normalization and test done in volumeShift function
[volumeChangePass, volumeChangeIndividualStatus, handles ] = volumeShift( volumeChangeSample, minReq4Band{ 2 }, handles );

if volumeChangePass == 1
    fprintf( 'This vehicle passes the relative volume change test\n');
    volumeChange = 'Pass';

else
```

```
    fprintf( 'This vehicle did not pass the 4-band relative volume change test\n');
    volumeChange = 'Fail';

end

% For logging
if handles.pass (2) == 1
    relVolStatusStat10 = { 'This vehicle passes the relative volume test for the Stationary and 10 kmh operating conditions ' } ;
else
    relVolStatusStat10 = { 'This vehicle does not pass the relative volume test for the Stationary and 10 kmh operating conditions ' } ;
end

if handles.pass (3) == 1
    relVolStatus10_20 = { 'This vehicle passes the relative volume test for the 10 kmh and 20 kmh operating conditions ' } ;
else
    relVolStatus10_20 = { 'This vehicle does not pass the relative volume test for the 10 kmh and 20 kmh operating conditions ' } ;
end

if handles.pass (4) == 1
    relVolStatus20_30 = { 'This vehicle passes the relative volume test for the 20 kmh and 30 kmh operating conditions ' } ;
else
    relVolStatus20_30 = { 'This vehicle does not pass the relative volume test for the 20 kmh and 30 kmh operating conditions ' } ;
end

relVolSignalLabelStat_10kmh = { 'Averaged Corrected SPL, Stationary, dBA' ; 'Averaged Corrected SPL, 10 kmh, dBA' } ;
relVolSignalLabel10kmh_20kmh = { 'Averaged Corrected SPL, 10 kmh, dBA' ; 'Averaged Corrected SPL, 20 kmh, dBA'  } ;
relVolSignalLabel20kmh_30kmh = { 'Averaged Corrected SPL, 20 kmh, dBA' ; 'Averaged Corrected SPL, 30 kmh, dBA' } ;

relVolMinReqLabel = { 'Minimum Requirement (Stationary, Section S5.1.1.2, Table 1), dBA' };

relVolNormalizedLabelStat_10kmh = { 'Normalized SPL = Stationary SPL - Requirement, dB' ; 'Normalized Signal = 10 kmh Level - Requirement, dB' } ;
relVolNormalizedLabel10kmh_20kmh = { 'Normalized SPL = 10 kmh SPL - Requirement, dB' ; 'Normalized Signal = 20 kmh Level - Requirement, dB' } ;
relVolNormalizedLabel20kmh_30kmh = { 'Normalized SPL = 20 kmh SPL - Requirement, dB' ; 'Normalized Signal = 30 kmh Level - Requirement, dB' } ;

relVolBandSumLabelStat_10kmh =  { 'Normalized Band Sum, Stationary, dBA' ; 'Normalized Band Sum, 10 kmh, dBA' } ;
relVolBandSumLabel10kmh_20kmh = { 'Normalized Band Sum, 10 kmh, dBA' ; 'Normalized Band Sum, 20 kmh, dBA' } ;
relVolBandSumLabel20kmh_30kmh = { 'Normalized Band Sum, 20 kmh, dBA' ; 'Normalized Band Sum, 30 kmh, dBA' } ;

relVolShiftLabelStat_10 = { 'Relative Volume Change Between Stationary and 10 kmh, dB' } ;
relVolShiftLabel10_20 = { 'Relative Volume Change Between 10kmh and 20 kmh, dB' } ;
relVolShiftLabel20_30 = { 'Relative Volume Change Between 20 kmh and 30 kmh, dB' } ;
```

```
481
482
483   relVolSignalStat = ( round( handles.meanDataSample{ 2 } * 10) / 10 ) ;
484   relVolSignal10kmh = ( round( handles.meanDataSample{ 3 } * 10) / 10 ) ;
485   relVolSignal20kmh = ( round( handles.meanDataSample{ 4 } * 10) / 10 ) ;
486   relVolSignal30kmh = ( round( handles.meanDataSample{ 5 } * 10) / 10 ) ;
487
488   relVolNormSignalStat =( round( handles.normalizedDataSample{ 2 } * 10) / 10 ) ;
489   relVolNormSignal10kmh =( round( handles.normalizedDataSample{ 3 } * 10) / 10 ) ;
490   relVolNormSignal20kmh =( round( handles.normalizedDataSample{ 4 } * 10) / 10 ) ;
491   relVolNormSignal30kmh =( round( handles.normalizedDataSample{ 5 } * 10) / 10 ) ;
492
493   relVolNormBandSumStat = ( round( handles.normalizedBandSum( 2 ) * 10) / 10 ) ;
494   relVolNormBandSum10kmh = ( round( handles.normalizedBandSum( 3 ) * 10) / 10 ) ;
495   relVolNormBandSum20kmh = ( round( handles.normalizedBandSum( 4 ) * 10) / 10 ) ;
496   relVolNormBandSum30kmh = ( round( handles.normalizedBandSum( 5 ) * 10) / 10 ) ;
497
498   relVolShiftStat_10kmh = ( round( handles.shift( 2 ) * 10) / 10 ) ;
499   relVolShift10kmh_20kmh = ( round( handles.shift( 3 ) * 10) / 10 ) ;
500   relVolShift20kmh_30kmh = ( round( handles.shift( 4 ) * 10) / 10 ) ;
501
502
503   %% Check Compliance
504   % ----------------------------------------------------------------
505   % For stop button
506       drawnow
507       if get(handles.pushbuttonStop, 'userdata') % stop condition
508           handles.stopButtonFlag = true ;
509           return;
510       end
511   % ----------------------------------------------------------------
512
513   set( handles.textProcessingStep, 'String', 'Writing Log Files' );
514   set( handles.displayCurrentOC, 'Visible', 'off' );
515   set( handles.textRunStatusFor, 'Visible', 'off' );
516   drawnow
517
518   %Initialize cells
519   micChoiceStatus = cell( 1, 5 );
520   samples_numbers = cell( 1, 5 );
521   samples_used = cell( 1, 5 );
522
523   for ink = 1 : 5
524       micChoiceStatus{ ink, 1 } = OC( ink );
525       micChoiceStatus{ ink, 2 } = micChoice{ ink };
526   end
527
528   micChoiceSimple = strrep( micChoice, 'Lower side: ', '' ) ;
529
530   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 ;...
    '10 km/h', ocPass( 3 ), '', ocPass2( 3 ), '' ; '20 km/h', ocPass( 4 ), '', ocPass2( 4 ), ''; '30 km/h', ocPass( 5 ), '', ocPass2( 5 ), '' };

status4BandDirectivityReverse = '-' ;
status2BandDirectivityReverse = '-' ;
status4BandDirectivity10kmh = '-' ;
status2BandDirectivity10kmh = '-' ;
status4BandDirectivity20kmh = '-' ;
status2BandDirectivity20kmh = '-' ;
status4BandDirectivity30kmh = '-' ;
status2BandDirectivity30kmh = '-' ;
relVolNormBandSumRev = '-' ;

relVolNormBandSumStatRound = num2str( ( round( relVolNormBandSumStat * 10) / 10 ) );
relVolNormBandSum10kmhRound = num2str( ( round( relVolNormBandSum10kmh * 10) / 10 ) ) ;
relVolNormBandSum20kmhRound = num2str( ( round( relVolNormBandSum20kmh * 10) / 10 ) );
relVolNormBandSum30kmhRound = num2str( ( round( relVolNormBandSum30kmh * 10) / 10 ) );

ocPassStr = arrayfun(@num2str,ocPass, 'un', 0 ) ;
ocPass2Str = arrayfun(@num2str,ocPass2, 'un', 0 ) ;
dPassStr = num2str( dPass ) ;
dPass2Str = num2str( dPass2 ) ;

handles.datStatus = [ statusMicSideReverse, statusMicSideStationary, statusMicSide10kmh, statusMicSide20kmh, statusMicSide30kmh ; ...
    ocPassStr( 1 ), ocPassStr( 2 ), ocPassStr( 3 ), ocPassStr( 4 ), ocPassStr( 5 ); ...
    { status4BandDirectivityReverse }, { dPassStr  }, { status4BandDirectivity10kmh }, { status4BandDirectivity20kmh }, { status4BandDirectivity30kmh }; ...
    ocPass2Str( 1 ), ocPass2Str( 2 ), ocPass2Str( 3 ), ocPass2Str( 4 ), ocPass2Str( 5 ); ...
    { status2BandDirectivityReverse }, { dPass2Str }, { status2BandDirectivity10kmh }, { status2BandDirectivity20kmh }, { status2BandDirectivity30kmh }; ...
    { relVolNormBandSumRev }, { relVolNormBandSumStatRound }, { relVolNormBandSum10kmhRound }, { relVolNormBandSum20kmhRound }, { relVolNormBandSum30kmhRound }] ;

% Assumes all tests fail first
passTestOC = [ 0 0 0 0 0 ];
passTestDir = 0;
passTestVol = 0;

% Passes the operating condition (OC) test if any OC passes EITHER the 2 OR 4 band test
```

```
cnt = 1;
for ink = 1 : 5
    if ocPass( ink ) || ocPass2( ink )
        passTestOC( ink ) = 1;
    else
        failsOC( cnt ) = ink;
        cnt = cnt + 1;
    end
end

% If any of the OCs fail BOTH the 2 and 4 band test, the vehicle fails
if any( passTestOC == 0 )
    passTestallOCs = 0 ;
else
    passTestallOCs = 1 ;
end

% Passes the directivity test if EITHER 2 OR 4 band directivity passes
if dPass || dPass2
    passTestDir = 1;
else
        failsDir( cnt ) = ink + 1;
        cnt = cnt + 1;
end

% Must pass relative volume (volume shift) test for 0kmh to 10kmh, 10kmh to 20kmh, AND 20kmh to 30kmh
if sum( volumeChangeIndividualStatus ) == 3
    passTestVol = 1;
else
    failsVol( cnt ) = ink + 2;
    cnt = cnt + 1;
end

% If any of the tests (OC, directivity, relative volume) fail, the vehicle
% is noncompliant
if passTestallOCs && passTestDir && passTestVol
    handles.complianceOverall = 'Pass' ;
else
    handles.complianceOverall = 'Fail' ;
end

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

```
opCondOptions = ({ 'Reverse', 'Stationary', '10kmh', '20kmh', '30kmh' }) ;

if reporting == 1 ;

    for ink = 1 : 5
        ocName = opCondOptions{ ink } ;
        yearStr = num2str( year ) ;

        % Vehicle and Test Site Info
        fileName = sprintf('%s%s%s_%s_%d.xlsx',year, make, model, ocName, numberInc );
        fileNamePath = fullfile( handles.outputDirectory, fileName) ;

        vehicle_info = { 'Vehicle Information', ''; 'Year', yearStr; 'Make', make; 'Model', model; 'NHTSA #',
        vehicleNHTSAnum; 'VIN', vehicleVIN };
        xlswrite( fileNamePath, vehicle_info, 'Sheet1', 'A5' );

        %Operating condition and lower side
        micOC = micChoiceSimple{ ink } ;
        ocMic_info = {'Operating Condition', ocName ; 'Lower Side', micOC };
        xlswrite( fileNamePath, ocMic_info, 'Sheet1', 'A12' );

        samples_tested_headers = { 'Samples Input for Testing ( Lower Side )' };
        samples_testedAmb_headers = { 'Samples Input for Testing ( Lower Side Ambient )' };
        samples_testedFront_headers = { 'Samples Input for Testing ( Front )' };
        samples_testedFrontAmb_headers = { 'Samples Input for Testing ( Front Ambient )' };
        samples_passing_2dB_headers = { 'Samples Used in 4 Band and 2 Band Tests ( Lower Side )' };
        samples_passing_2dB_headers_amb = { 'Samples Used in 4 Band and 2 Band Tests ( Lower Side Ambient )' };
        samples_passing_2dB_headers_front = { 'Samples Used in 4 Band and 2 Band Tests ( Front )' };
        samples_passing_2dB_headers_frontAmb = { 'Samples Used in 4 Band and 2 Band Tests ( Front Ambient )' };

        sheet1Headers = [samples_tested_headers ; ' ' ; samples_testedAmb_headers; ' '; ' ';
        samples_testedFront_headers; ' '; ...
            samples_testedFrontAmb_headers; ' '; ' '; samples_passing_2dB_headers; ' ';
            samples_passing_2dB_headers_amb; ' '; ' ';...
            samples_passing_2dB_headers_front; ' '; samples_passing_2dB_headers_frontAmb ];

        % write sheet 1 headers
        xlswrite( fileNamePath, sheet1Headers, 'Sheet1', 'A14' );

        % Lower side signal files
        xlswrite( fileNamePath, fileNames.side { ink }, 'Sheet1', 'B14' );

        % Lower side ambient
        xlswrite( fileNamePath, fileNames.sideambient { ink }, 'Sheet1', 'B16' );
        xlswrite( fileNamePath, fileNames.ambientSide { ink }, 'Sheet1', 'B17' );

        % Front side signal files
        xlswrite( fileNamePath, { 'NA' }, 'Sheet1', 'B19' );
```

```
671         % Front ambient files
672         xlswrite( fileNamePath, { 'NA' }, 'Sheet1', 'B21' );
673
674         if ink == 2
675             xlswrite( fileNamePath, fileNames.front { ink }, 'Sheet1', 'B19' );
676             xlswrite( fileNamePath, fileNames.frontambient { ink }, 'Sheet1', 'B21' );
677             xlswrite( fileNamePath, fileNames.frontambientSide { ink }, 'Sheet1', 'B22' );
678         end
679
680         % Samples used in 4 band and 2 band tests (lower side)
681         xlswrite( fileNamePath, fileNames.side4 { ink }, 'Sheet1', 'B24' );
682
683         % Samples used in 4 band and 2 band tests lower side ambient
684         xlswrite( fileNamePath, fileNames.sideambient4 { ink }, 'Sheet1', 'B26' );
685         xlswrite( fileNamePath, fileNames.ambientSide4 { ink }, 'Sheet1', 'B27' );
686
687         %Samples used in 2 band and 4 band - Front
688         xlswrite( fileNamePath, { 'NA' }, 'Sheet1', 'B29' );
689
690         % Samples used in 2 band and 4 band - Front ambient
691         xlswrite( fileNamePath, { 'NA' }, 'Sheet1', 'B31' );
692
693         if ink == 2
694             xlswrite( fileNamePath, fileNames.front4 { ink }, 'Sheet1', 'B29' );
695             xlswrite( fileNamePath, fileNames.frontambient4 { ink }, 'Sheet1', 'B31' );
696             xlswrite( fileNamePath, fileNames.frontambientSide4 { ink }, 'Sheet1', 'B32' );
697         end
698
699         idxvalidTrue = cell( 1, 5 );
700
701         headers_4Band2Band = {'4 Band Test', '', '', '', '', '', '' ,'' ,'' ,'' ,'', '2 Band Test';} ;
702         headers_OCDirectivity = { 'Bands (Hz)', 'Averaged Corrected SPL (dBA)', 'Minimum Requirement (Section
            S5.1.1.2, Table 1) (dBA)', 'Difference, dB', '', '4-Band Combination', '', '', '', 'Pass=1/Fail=0', '',
            'Bands', 'Averaged Corrected SPL (dBA)', 'Minimum Requirement (Section S5.2, Table 6) (dBA)', 'Difference,
            dB','', '', '2-Band Combination', '', 'Band Sum for Passing Combinations', 'Pass=1/Fail=0';} ;
703         relVolSheetHeader = {' Relative Volume Results' } ;
704         header_highestBelow1k = {'Highest level below 1000 Hz'};
705         header_highest1kAndUp = {'Highest level between 1000 Hz and 3150 Hz'};
706         header_secondHighestBelow1k = {'Second highest level below 1000 Hz'};
707         header_secondHighest1kAndUp = {'Second highest level between 1000 Hz and 3150 Hz'};
708         bandsNumb = [ 315; 400; 500; 630; 800; 1000; 1250; 1600; 2000; 2500; 3150; 4000; 5000 ] ;
709         bandsNumb2Band = [ 315; 400; 500; 630; 800; 1000; 1250; 1600; 2000; 2500; 3150 ] ;
710
711         if ink == 1 ;
712             % Reverse Operating Condition
713             reverseSheetHeader = {'Reverse Condition Samples - Band Combinations' } ;
714
715             % 4 band test
716             fourBandSignalReqDiff = [bandsNumb, NHTSAsampleData{ ink }' , minReq4Band{ ink }', (NHTSAsampleData{ ink
                }' - minReq4Band{ ink }')] ;
```

```
717
718             % 2 band test
719             % Find highest and second highest levels and the corresponding excel cell
720             twoBandSignalReqDiff = [bandsNumb2Band, allianceSampleData{ ink }' , minReqBandMinValbyBand{ ink }',
                (allianceSampleData{ ink }' - minReqBandMinValbyBand{ ink }')] ;
721
722             allianceBelow1k = allianceSampleData{ink}(:, 1:5);
723             alliance1kandUp = allianceSampleData{ink}(:, 6:11);
724             [sortBelow1k, idxBelow1k] = sort(allianceBelow1k, 'descend');
725             [sort1kAndUp, idx1kAndUp] = sort(alliance1kandUp, 'descend');
726
727             cell_highestBelow1k = strcat('P', num2str( idxBelow1k(1) + 6 ) );
728             cell_highest1kAndUp = strcat('P', num2str( idx1kAndUp(1) + 11 ) );
729             cell_secondHighestBelow1k = strcat('P', num2str( idxBelow1k(2) + 6 ) );
730             cell_secondHighest1kAndUp = strcat('P', num2str( idx1kAndUp(2) + 11 ) );
731
732             % Overall status for OC test
733                 if passFail == 5 && passFail2 == 5
734                     ocTestStatus = { 'This vehicle passes the operating condition test for both the 4-band test and
                        the 2-band test' } ;
735                 elseif passFail == 5
736                     ocTestStatus = { 'This vehicle passes the operating condition test for the 4-band test but not
                        for the 2-band test' } ;
737                 elseif passFail2 ==5
738                     ocTestStatus = { 'This vehicle passes the operating condition test for the 2-band test but not
                        for the 4-band test' } ;
739                 else
740                     ocTestStatus = { 'This vehicle does not pass the operating condition test for either the 2-band
                        test or the 4-band test' } ;
741                 end
742
743             % Write to log
744             xlswrite( fileNamePath, ocTestStatus, 'Reverse', 'A3' );
745             xlswrite( fileNamePath, headers_4Band2Band, 'Reverse', 'A5' );
746             xlswrite( fileNamePath, headers_OCDirectivity, 'Reverse', 'A6' );
747
748             %4 band
749             xlswrite( fileNamePath, fourBandSignalReqDiff, 'Reverse', 'A7' );
750             xlswrite( fileNamePath, results_4band{ ink }, 'Reverse', 'F7' );
751
752             %2 band
753             xlswrite( fileNamePath, twoBandSignalReqDiff, 'Reverse', 'L7' );
754             xlswrite( fileNamePath, results_2band{ ink }, 'Reverse', 'R7' );
755             xlswrite( fileNamePath, header_highestBelow1k, 'Reverse', cell_highestBelow1k );
756             xlswrite( fileNamePath, header_highest1kAndUp, 'Reverse', cell_highest1kAndUp  );
757             xlswrite( fileNamePath, header_secondHighestBelow1k, 'Reverse', cell_secondHighestBelow1k );
758             xlswrite( fileNamePath, header_secondHighest1kAndUp, 'Reverse', cell_secondHighest1kAndUp  );
759
760             xlswrite( fileNamePath, reverseSheetHeader, 'Reverse', 'A1' );
761
```

```
        end

         if ink == 2 ;
             % Stationary Operating Condition
             StationarySheetHeader = {'Stationary Condition Samples - Band Combinations' } ;
             DirectivitySheetHeader = {'Directivity Results' } ;

             % 4 band test
             fourBandSignalReqDiff = [bandsNumb, NHTSAsampleData{ ink }' , minReq4Band{ ink }', (NHTSAsampleData{ ink }' - minReq4Band{ ink }')] ;

             % 2 band test
             % Find highest and second highest levels and the corresponding excel cell
             twoBandSignalReqDiff = [bandsNumb2Band, allianceSampleData{ ink }' , minReqBandMinValbyBand{ ink }', (allianceSampleData{ ink }' - minReqBandMinValbyBand{ ink }')] ;

             allianceBelow1k = allianceSampleData{ink}(:, 1:5);
             alliance1kandUp = allianceSampleData{ink}(:, 6:11);
             [sortBelow1k, idxBelow1k] = sort(allianceBelow1k, 'descend');
             [sort1kAndUp, idx1kAndUp] = sort(alliance1kandUp, 'descend');

             cell_highestBelow1k = strcat('P', num2str( idxBelow1k(1) + 6 ) );
             cell_highest1kAndUp = strcat('P', num2str( idx1kAndUp(1) + 11 ) );
             cell_secondHighestBelow1k = strcat('P', num2str( idxBelow1k(2) + 6 ) );
             cell_secondHighest1kAndUp = strcat('P', num2str( idx1kAndUp(2) + 11 ) );

             % Write to log
             xlswrite( fileNamePath, headers_4Band2Band, 'Stationary', 'A5' );
             xlswrite( fileNamePath, headers_OCDirectivity, 'Stationary', 'A6' );

             %4 band
             xlswrite( fileNamePath, fourBandSignalReqDiff, 'Stationary', 'A7' );
             xlswrite( fileNamePath, results_4band{ ink }, 'Stationary', 'F7' );

             %2 band
             xlswrite( fileNamePath, twoBandSignalReqDiff, 'Stationary', 'L7' );
             xlswrite( fileNamePath, results_2band{ ink }, 'Stationary', 'R7' );
             xlswrite( fileNamePath, header_highestBelow1k, 'Stationary', cell_highestBelow1k );
             xlswrite( fileNamePath, header_highest1kAndUp, 'Stationary', cell_highest1kAndUp  );
             xlswrite( fileNamePath, header_secondHighestBelow1k, 'Stationary', cell_secondHighestBelow1k );
             xlswrite( fileNamePath, header_secondHighest1kAndUp, 'Stationary', cell_secondHighest1kAndUp  );

             xlswrite( fileNamePath, StationarySheetHeader, 'Stationary', 'A1' );

             % Directivity log sheet
             xlswrite( fileNamePath, directivityStatus, 'Directivity', 'A3' );

             directivity4Band = [ fourBandCombos, passFailStatus4Band ] ;
             directivity2Band = [ twoBandCombos, bandSumTrackingDirectivity', passFailStatus2Band ] ;

```

```
            % 4 band test
            fourBandSignalReqDiff = [bandsNumb, stationarySignal' , minReq4Band{ ink }', (NHTSAsampleData{ ink }' - minReq4Band{ ink }')] ;

            % 2 band test
            % Find highest and second highest levels and the corresponding excel cell
            twoBandSignalReqDiff = [bandsNumb2Band, stationarySignal(:,1:11)' , minReqBandMinValbyBand{ ink }', ( stationarySignal( :, 1:11 )' - minReqBandMinValbyBand{ ink }' ) ] ;

            allianceBelow1kFront = stationarySignal(:, 1:5);
            alliance1kandUpFront = stationarySignal(:, 6:11);
            [sortBelow1k, idxBelow1k] = sort(allianceBelow1kFront, 'descend');
            [sort1kAndUp, idx1kAndUp] = sort(alliance1kandUpFront, 'descend');

            cell_highestBelow1k = strcat('P', num2str( idxBelow1k(1) + 6 ) );
            cell_highest1kAndUp = strcat('P', num2str( idx1kAndUp(1) + 11 ) );
            cell_secondHighestBelow1k = strcat('P', num2str( idxBelow1k(2) + 6 ) );
            cell_secondHighest1kAndUp = strcat('P', num2str( idx1kAndUp(2) + 11 ) );

            % Write to log
            xlswrite( fileNamePath, headers_4Band2Band, 'Directivity', 'A5' );
            xlswrite( fileNamePath, headers_OCDirectivity, 'Directivity', 'A6' );

            %4 band
            xlswrite( fileNamePath, fourBandSignalReqDiff, 'Directivity', 'A7' );
            xlswrite( fileNamePath, directivity4Band, 'Directivity', 'F7' );

            %2 band
            xlswrite( fileNamePath, twoBandSignalReqDiff, 'Directivity', 'L7' );
            xlswrite( fileNamePath, directivity2Band, 'Directivity', 'R7' );
            xlswrite( fileNamePath, header_highestBelow1k, 'Directivity', cell_highestBelow1k );
            xlswrite( fileNamePath, header_highest1kAndUp, 'Directivity', cell_highest1kAndUp  );
            xlswrite( fileNamePath, header_secondHighestBelow1k, 'Directivity', cell_secondHighestBelow1k );
            xlswrite( fileNamePath, header_secondHighest1kAndUp, 'Directivity', cell_secondHighest1kAndUp  );


            xlswrite( fileNamePath, DirectivitySheetHeader, 'Directivity', 'A1' );

        end

        if ink == 3 ;

            % 10 kph Operating Condition
            tenkmhSheetHeader = {'10 kmh Condition Samples - Band Combinations' } ;

            % 4 band test
            fourBandSignalReqDiff = [bandsNumb, NHTSAsampleData{ ink }' , minReq4Band{ ink }', (NHTSAsampleData{ ink }' - minReq4Band{ ink }')] ;

            % 2 band test
```

```
857                 % Find highest and second highest levels and the corresponding excel cell
858                 twoBandSignalReqDiff = [bandsNumb2Band, allianceSampleData{ ink }' , minReqBandMinValbyBand{ ink }',
                    (allianceSampleData{ ink }' - minReqBandMinValbyBand{ ink }')] ;
859
860                 allianceBelow1k = allianceSampleData{ink}(:, 1:5);
861                 alliance1kandUp = allianceSampleData{ink}(:, 6:11);
862                 [sortBelow1k, idxBelow1k] = sort(allianceBelow1k, 'descend');
863                 [sort1kAndUp, idx1kAndUp] = sort(alliance1kandUp, 'descend');
864
865                 cell_highestBelow1k = strcat('P', num2str( idxBelow1k(1) + 6 ) );
866                 cell_highest1kAndUp = strcat('P', num2str( idx1kAndUp(1) + 11 ) );
867                 cell_secondHighestBelow1k = strcat('P', num2str( idxBelow1k(2) + 6 ) );
868                 cell_secondHighest1kAndUp = strcat('P', num2str( idx1kAndUp(2) + 11 ) );
869
870                 % Write to log
871                 xlswrite( fileNamePath, headers_4Band2Band, '10_kmh', 'A5' );
872                 xlswrite( fileNamePath, headers_OCDirectivity, '10_kmh', 'A6' );
873
874                 %4 band
875                 xlswrite( fileNamePath, fourBandSignalReqDiff, '10_kmh', 'A7' );
876                 xlswrite( fileNamePath, results_4band{ ink }, '10_kmh', 'F7' );
877
878                 %2 band
879                 xlswrite( fileNamePath, twoBandSignalReqDiff, '10_kmh', 'L7' );
880                 xlswrite( fileNamePath, results_2band{ ink }, '10_kmh', 'R7' );
881                 xlswrite( fileNamePath, header_highestBelow1k, '10_kmh', cell_highestBelow1k );
882                 xlswrite( fileNamePath, header_highest1kAndUp, '10_kmh', cell_highest1kAndUp  );
883                 xlswrite( fileNamePath, header_secondHighestBelow1k, '10_kmh', cell_secondHighestBelow1k );
884                 xlswrite( fileNamePath, header_secondHighest1kAndUp, '10_kmh', cell_secondHighest1kAndUp  );
885
886                 xlswrite( fileNamePath, tenkmhSheetHeader, '10_kmh', 'A1' );
887
888                 % Relative volume log sheet
889                 xlswrite( fileNamePath, bands, 'RelativeVolume', 'B5' );
890                 xlswrite( fileNamePath, relVolStatusStat10, 'RelativeVolume', 'A3' );
891                 xlswrite( fileNamePath, relVolSignalLabelStat_10kmh, 'RelativeVolume', 'A6' );
892                 xlswrite( fileNamePath, relVolSignalStat, 'RelativeVolume', 'B6' );
893                 xlswrite( fileNamePath, relVolSignal10kmh, 'RelativeVolume', 'B7' );
894
895                 xlswrite( fileNamePath, relVolMinReqLabel, 'RelativeVolume', 'A9' );
896                 xlswrite( fileNamePath, handles.RelVolMinReq, 'RelativeVolume', 'B9' );
897
898                 xlswrite( fileNamePath, relVolNormalizedLabelStat_10kmh, 'RelativeVolume', 'A11' );
899                 xlswrite( fileNamePath, relVolNormSignalStat, 'RelativeVolume', 'B11' );
900                 xlswrite( fileNamePath, relVolNormSignal10kmh, 'RelativeVolume', 'B12' );
901
902                 xlswrite( fileNamePath, relVolBandSumLabelStat_10kmh, 'RelativeVolume', 'A14' );
903                 xlswrite( fileNamePath, relVolNormBandSumStat, 'RelativeVolume', 'B14' );
904                 xlswrite( fileNamePath, relVolNormBandSum10kmh, 'RelativeVolume', 'B15' );
905
```

```
            xlswrite( fileNamePath, relVolShiftLabelStat_10, 'RelativeVolume', 'A17' );
            xlswrite( fileNamePath, relVolShiftStat_10kmh, 'RelativeVolume', 'B17' );

            xlswrite( fileNamePath, relVolSheetHeader, 'RelativeVolume', 'A1' );


        end

        if ink == 4 ;

            % 20 kph Operating Condition
            twentykmhSheetHeader = {'20 kmh Condition Samples - Band Combinations' } ;

            % 4 band test
            fourBandSignalReqDiff = [bandsNumb, NHTSAsampleData{ ink }' , minReq4Band{ ink }', (NHTSAsampleData{ ink }' - minReq4Band{ ink }')] ;

            % 2 band test
            % Find highest and second highest levels and the corresponding excel cell
            twoBandSignalReqDiff = [bandsNumb2Band, allianceSampleData{ ink }' , minReqBandMinValbyBand{ ink }', (allianceSampleData{ ink }' - minReqBandMinValbyBand{ ink }')] ;

            allianceBelow1k = allianceSampleData{ink}(:, 1:5);
            alliance1kandUp = allianceSampleData{ink}(:, 6:11);
            [sortBelow1k, idxBelow1k] = sort(allianceBelow1k, 'descend');
            [sort1kAndUp, idx1kAndUp] = sort(alliance1kandUp, 'descend');

            cell_highestBelow1k = strcat('P', num2str( idxBelow1k(1) + 6 ) );
            cell_highest1kAndUp = strcat('P', num2str( idx1kAndUp(1) + 11 ) );
            cell_secondHighestBelow1k = strcat('P', num2str( idxBelow1k(2) + 6 ) );
            cell_secondHighest1kAndUp = strcat('P', num2str( idx1kAndUp(2) + 11 ) );

            % Write to log
            xlswrite( fileNamePath, headers_4Band2Band, '20_kmh', 'A5' );
            xlswrite( fileNamePath, headers_OCDirectivity, '20_kmh', 'A6' );

            %4 band
            xlswrite( fileNamePath, fourBandSignalReqDiff, '20_kmh', 'A7' );
            xlswrite( fileNamePath, results_4band{ ink }, '20_kmh', 'F7' );

            %2 band
            xlswrite( fileNamePath, twoBandSignalReqDiff, '20_kmh', 'L7' );
            xlswrite( fileNamePath, results_2band{ ink }, '20_kmh', 'R7' );
            xlswrite( fileNamePath, header_highestBelow1k, '20_kmh', cell_highestBelow1k );
            xlswrite( fileNamePath, header_highest1kAndUp, '20_kmh', cell_highest1kAndUp  );
            xlswrite( fileNamePath, header_secondHighestBelow1k, '20_kmh', cell_secondHighestBelow1k );
            xlswrite( fileNamePath, header_secondHighest1kAndUp, '20_kmh', cell_secondHighest1kAndUp  );

            xlswrite( fileNamePath, twentykmhSheetHeader, '20_kmh', 'A1' );

```

```
954             % Relative volume log sheet
955             xlswrite( fileNamePath, bands, 'RelativeVolume', 'B5' );
956             xlswrite( fileNamePath, relVolStatus10_20, 'RelativeVolume', 'A3' );
957             xlswrite( fileNamePath, relVolSignalLabel10kmh_20kmh, 'RelativeVolume', 'A6' );
958             xlswrite( fileNamePath, relVolSignal10kmh, 'RelativeVolume', 'B6' );
959             xlswrite( fileNamePath, relVolSignal20kmh, 'RelativeVolume', 'B7' );
960
961             xlswrite( fileNamePath, relVolMinReqLabel, 'RelativeVolume', 'A9' );
962             xlswrite( fileNamePath, handles.RelVolMinReq, 'RelativeVolume', 'B9' );
963
964             xlswrite( fileNamePath, relVolNormalizedLabel10kmh_20kmh, 'RelativeVolume', 'A11' );
965             xlswrite( fileNamePath, relVolNormSignal10kmh, 'RelativeVolume', 'B11' );
966             xlswrite( fileNamePath, relVolNormSignal20kmh, 'RelativeVolume', 'B12' );
967
968             xlswrite( fileNamePath, relVolBandSumLabel10kmh_20kmh, 'RelativeVolume', 'A14' );
969             xlswrite( fileNamePath, relVolNormBandSum10kmh, 'RelativeVolume', 'B14' );
970             xlswrite( fileNamePath, relVolNormBandSum20kmh, 'RelativeVolume', 'B15' );
971
972             xlswrite( fileNamePath, relVolShiftLabel10_20, 'RelativeVolume', 'A17' );
973             xlswrite( fileNamePath, relVolShift10kmh_20kmh, 'RelativeVolume', 'B17' );
974
975             xlswrite( fileNamePath, relVolSheetHeader, 'RelativeVolume', 'A1' );
976
977         end
978
979         if ink == 5 ;
980
981             % 30 kph Operating Condition
982             thirtykmhSheetHeader = strcat('30 kmh Condition Samples - Band Combinations for', {' '}, year, {' '},
                make, {' '}, model, ',', {' '}, handles.opCondSimple, ' operating condition') ;
983
984             % 4 band test
985             fourBandSignalReqDiff = [bandsNumb, NHTSAsampleData{ ink }' , minReq4Band{ ink }', (NHTSAsampleData{ ink
                }' - minReq4Band{ ink }')] ;
986
987             % 2 band test
988             % Find highest and second highest levels and the corresponding excel cell
989             twoBandSignalReqDiff = [bandsNumb2Band, allianceSampleData{ ink }' , minReqBandMinValbyBand{ ink }',
                (allianceSampleData{ ink }' - minReqBandMinValbyBand{ ink }')] ;
990
991             allianceBelow1k = allianceSampleData{ink}(:, 1:5);
992             alliance1kandUp = allianceSampleData{ink}(:, 6:11);
993             [sortBelow1k, idxBelow1k] = sort(allianceBelow1k, 'descend');
994             [sort1kAndUp, idx1kAndUp] = sort(alliance1kandUp, 'descend');
995
996             cell_highestBelow1k = strcat('P', num2str( idxBelow1k(1) + 6 ) );
997             cell_highest1kAndUp = strcat('P', num2str( idx1kAndUp(1) + 11 ) );
998             cell_secondHighestBelow1k = strcat('P', num2str( idxBelow1k(2) + 6 ) );
999             cell_secondHighest1kAndUp = strcat('P', num2str( idx1kAndUp(2) + 11 ) );
1000
```

```
            % Write to log
            xlswrite( fileNamePath, headers_4Band2Band, '30_kmh', 'A5' );
            xlswrite( fileNamePath, headers_OCDirectivity, '30_kmh', 'A6' );

            %4 band
            xlswrite( fileNamePath, fourBandSignalReqDiff, '30_kmh', 'A7' );
            xlswrite( fileNamePath, results_4band{ ink }, '30_kmh', 'F7' );

            %2 band
            xlswrite( fileNamePath, twoBandSignalReqDiff, '30_kmh', 'L7' );
            xlswrite( fileNamePath, results_2band{ ink }, '30_kmh', 'R7' );
            xlswrite( fileNamePath, header_highestBelow1k, '30_kmh', cell_highestBelow1k );
            xlswrite( fileNamePath, header_highest1kAndUp, '30_kmh', cell_highest1kAndUp  );
            xlswrite( fileNamePath, header_secondHighestBelow1k, '30_kmh', cell_secondHighestBelow1k );
            xlswrite( fileNamePath, header_secondHighest1kAndUp, '30_kmh', cell_secondHighest1kAndUp  );

            xlswrite( fileNamePath, thirtykmhSheetHeader, '30_kmh', 'A1' );

            % Relative volume log sheet
            xlswrite( fileNamePath, bands, 'RelativeVolume', 'B5' );
            xlswrite( fileNamePath, relVolStatus20_30, 'RelativeVolume', 'A3' );
            xlswrite( fileNamePath, relVolSignalLabel20kmh_30kmh, 'RelativeVolume', 'A6' );
            xlswrite( fileNamePath, relVolSignal20kmh, 'RelativeVolume', 'B6' );
            xlswrite( fileNamePath, relVolSignal30kmh, 'RelativeVolume', 'B7' );

            xlswrite( fileNamePath, relVolMinReqLabel, 'RelativeVolume', 'A9' );
            xlswrite( fileNamePath, handles.RelVolMinReq, 'RelativeVolume', 'B9' );

            xlswrite( fileNamePath, relVolNormalizedLabel20kmh_30kmh, 'RelativeVolume', 'A11' );
            xlswrite( fileNamePath, relVolNormSignal20kmh, 'RelativeVolume', 'B11' );
            xlswrite( fileNamePath, relVolNormSignal30kmh, 'RelativeVolume', 'B12' );

            xlswrite( fileNamePath, relVolBandSumLabel20kmh_30kmh, 'RelativeVolume', 'A14' );
            xlswrite( fileNamePath, relVolNormBandSum20kmh, 'RelativeVolume', 'B14' );
            xlswrite( fileNamePath, relVolNormBandSum30kmh, 'RelativeVolume', 'B15' );

            xlswrite( fileNamePath, relVolShiftLabel20_30, 'RelativeVolume', 'A17' );
            xlswrite( fileNamePath, relVolShift20kmh_30kmh, 'RelativeVolume', 'B17' );

            xlswrite( fileNamePath, relVolSheetHeader, 'RelativeVolume', 'A1' );

        end

        fileName = sprintf('%s%s%s_%s_%d.xlsx',year, make, model, ocName, numberInc );
        fileNamePath = fullfile( handles.outputDirectory, fileName) ;

        vehicle_info = { 'Vehicle Information', ''; 'Year', yearStr; 'Make', make; 'Model', model; 'NHTSA #',
        vehicleNHTSAnum; 'VIN', vehicleVIN };
        xlswrite( fileNamePath, vehicle_info, 'Sheet1', 'A5' );

```

```
1050            introHeader1 = {'Minimum Sound Requirements for Hybrid and Electric Vehicles' } ;
1051            introHeader21 = 'Compliance Tool Version ' ;
1052            introHeader22 = handles.versionNumber ;
1053            introHeader2 = { sprintf('%s%s', introHeader21, introHeader22)} ;
1054            introHeader3 = {'Log File'} ;
1055
1056            xlswrite (fileNamePath,introHeader1, 'Sheet1', 'A1')
1057            xlswrite (fileNamePath,introHeader2, 'Sheet1', 'A2')
1058            xlswrite (fileNamePath,introHeader3, 'Sheet1', 'A3')
1059
1060        end
1061
1062    %% Prep variables for summary log
1063
1064    vehicleInfo = [{ year } ; { make } ; { model } ; { vehicleNHTSAnum } ; { vehicleVIN } ] ;
1065    compliaceOverallResult = { handles.complianceOverall } ;
1066
1067    if passTestDir == 1
1068        passTestDirPF = 'Pass' ;
1069    else
1070        passTestDirPF = 'Fail' ;
1071    end
1072
1073    for ink = 1 : 5
1074        if ocPass( ink ) > 0
1075            ocPassStatus{ ink } = 'Pass';
1076        else
1077            ocPassStatus{ ink } = 'Fail';
1078        end
1079    end
1080
1081    for ink = 1 : 5
1082        if ocPass2( ink ) > 0
1083            ocPassStatus2{ ink } = 'Pass';
1084        else
1085            ocPassStatus2{ ink } = 'Fail';
1086        end
1087    end
1088
1089    for ink = 1 : 5
1090        if ocPass( ink ) || ocPass2( ink )
1091            ocPassOverall{ ink } = 'Pass';
1092        else
1093            ocPassOverall{ ink } = 'Fail';
1094        end
1095    end
1096
1097    fourFileNamesReverse = fileNames.side4 { 1 } ;
1098    fourFileNamesStationary = fileNames.side4 { 2 } ;
1099    fourFileNames10kmh = fileNames.side4 { 3 } ;
```

```
fourFileNames20kmh = fileNames.side4 { 4 } ;
fourFileNames30kmh = fileNames.side4 {5 } ;

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
```