```
%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 = '2.0.2.0';
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
```

```
    handles = dataImportBatch( handles ) ; % Imports metadata, not time signals, has a place holder for the time signals

    setappdata( 0, 'fileLoadCondition', handles.fileLoadCondition )
    setappdata( 0, 'datStationary', handles.datStationary )
    setappdata( 0, 'dat10kmh', handles.dat10kmh )
    setappdata( 0, 'dat20kmh', handles.dat20kmh )
    setappdata( 0, 'dat30kmh', handles.dat30kmh )
    setappdata( 0, 'datReverse', handles.datReverse )

    handles = GUI_fileLoad2( handles );

end

% --- Executes on selection change in controlOperatingCondition.
function controlOperatingCondition_Callback(hObject, eventdata, handles)
% hObject    handle to controlOperatingCondition (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 controlOperatingCondition contents as cell array
%        contents{get(hObject,'Value')} returns selected item from controlOperatingCondition

contents = cellstr( get( hObject, 'String' ) ); % Need to get contents, before you can index into them
handles.operatingCondition = contents{ get( hObject, 'Value' ) };

switch handles.operatingCondition
    case ' Stationary'
        handles.opCond = 'Stationary';
    case ' 10 km/h'
        handles.opCond = '10 km/h';
    case ' 20 km/h'
        handles.opCond = '20 km/h';
    case ' 30 km/h'
        handles.opCond = '30 km/h';
    case ' Reverse'
        handles.opCond = 'Reverse';
end

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

handles.currentSample = 1;

if ~ isempty( handles.data )

    if ~ isempty( handles.data( handles.data.opCond == handles.opCond, : ) )

        %clear all displays
        set( handles.displayDriverSignalFile,         'String', '') ;
```

```
        set( handles.displayDriverPreAmbientFile,     'String', '') ;
        set( handles.displayDriverPostAmbientFile,    'String', '') ;
        set( handles.displayPassengerSignalFile,      'String', '') ;
        set( handles.displayPassengerPreAmbientFile,  'String', '') ;
        set( handles.displayPassengerPostAmbientFile, 'String', '') ;
        set( handles.displayTriggerFile,              'String', '') ;
        set( handles.editTriggerThreshold,            'String', '') ;
        set( handles.displayFrontMicSignalFile,       'String', '') ;
        set( handles.displayFrontMicPreAmbientFile,   'String', '') ;
        set( handles.displayFrontMicPostAmbientFile,  'String', '') ;

        %Enable displays according to OC
        if strcmpi( handles.opCond, 'Stationary' );
            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', 'on' );
            set( handles.textFrSig,                        'Enable', 'on' );
            set( handles.displayFrontMicSignalFile,        'Enable', 'on' );
            set( handles.controlFrontMicSignalChannel,     'Enable', 'on' );
            set( handles.textFrPreAmb,                     'Enable', 'on' );
            set( handles.textFrPostAmb,                    'Enable', 'on' );
            set( handles.displayFrontMicPreAmbientFile,    'Enable', 'on' );
            set( handles.displayFrontMicPostAmbientFile,   'Enable', 'on' );
            set( handles.controlFrontMicPreAmbientChannel, 'Enable', 'on' );
            set( handles.controlFrontMicPostAmbientChannel, 'Enable', 'on' );

        elseif  strcmpi( handles.opCond, '10 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, '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 handles.data.opCond( ink ) == handles.opCond;
        if  strcmp( char( handles.data{ ink, 'signalType' } ), 'trigger' ) && handles.data.sampleID( ink ) == handles.sampleID ;
            handles.data( ink, 'TriggerThreshold' ) = replacedata( handles.data( ink, 'TriggerThreshold' ), {handles.triggerThresholdValue} );
        end
    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
```

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

```
    mkdir( currentRunOutputFolder );
end

% 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' ) )
set( handles.controlPlayDriverSignal,             'Enable', 'off' )
set( handles.controlPlayPassengerSignal,          'Enable', 'off' )
set( handles.controlPlayAmbientimeTriggerSignal, 'Enable', 'off' )
set( handles.uitable2,                            'Data', [] )

function handles = plotNextPreviousSamples( handles )
% This code is used for next, discard, and previous

% Turn off second y axis between plots
if isfield( handles, 'ax' )
    if length ( handles.ax ) > 1
        set( handles.ax( 2 ), 'Visible' , 'Off' )
        cla( handles.ax( 2 ) )
    end
end

if ~ isempty( handles.data )
    if ~ isempty( handles.data( handles.data.opCond == handles.opCond, : ) )

        % Clear all displays
        set( handles.displayDriverSignalFile,        'String', '') ;
        set( handles.displayDriverPreAmbientFile,    'String', '') ;
        set( handles.displayPassengerSignalFile,     'String', '') ;
        set( handles.displayPassengerPreAmbientFile, 'String', '') ;
        set( handles.displayTriggerFile,             'String', '') ;
        set( handles.displayFrontMicSignalFile,      'String', '') ;
        set( handles.displayFrontMicPreAmbientFile,  'String', '') ;

        % Enable displays according to OC
        if strcmpi( handles.opCond, 'Stationary' );
            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', 'on' );
            set( handles.textFrSig,                        'Enable', 'on' );
            set( handles.displayFrontMicSignalFile,        'Enable', 'on' );
            set( handles.controlFrontMicSignalChannel,     'Enable', 'on' );
            set( handles.textFrPreAmb,                     'Enable', 'on' );
```

```
            set( handles.displayFrontMicPreAmbientFile,    'Enable', 'on' );
            set( handles.controlFrontMicPreAmbientChannel, 'Enable', 'on' );

        elseif  strcmpi( handles.opCond, '10 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, '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.displayFrontMicPreAmbientFile,    'Enable', 'off' );
            set( handles.controlFrontMicPreAmbientChannel, '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.displayFrontMicPreAmbientFile,    'Enable', 'off' );
            set( handles.controlFrontMicPreAmbientChannel, 'Enable', 'off' );
        end

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

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

        handles = loadData( handles );
        handles = plotData( handles );

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

```
function handles = loadData( handles )

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 );
set( handles.controlDriverPreAmbientChannel, 'Enable', 'off' );

%-------------------------------------------------------------------------
% Driver Post-ambient
%-------------------------------------------------------------------------
handles.driverPostAmbientPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'driver' & handles.data.signalType == 'postAmbient' & ...
    handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
handles.driverPostAmbientFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'driver' & handles.data.signalType == 'postAmbient' & ...
    handles.data.sampleID == handles.sampleID, { 'fileName' } ) );

set( handles.displayDriverPostAmbientFile, 'String', [ handles.driverPostAmbientPathName{ 1 } handles.driverPostAmbientFileName{ 1 } ] );
set( handles.displayDriverPostAmbientFile, 'Enable', 'on' );

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

[ handles.driverPostAmbient_Amp, handles.driverAFs ] = wavread( [ handles.driverPostAmbientPathName{ 1 } handles.driverPostAmbientFileName{ 1 } ] );
handles.driverPostAmbient_Amp = handles.driverPostAmbient_Amp( :, driverPostAmbientChannel );

set( handles.controlDriverPostAmbientChannel, 'Value', driverPostAmbientChannel );
set( handles.controlDriverPostAmbientChannel, 'Enable', 'off' );

%-------------------------------------------------------------------------
% Passenger Signal
%-------------------------------------------------------------------------
handles.passengerSignalPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'passenger' & handles.data.signalType == 'signal' & ...
    handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
handles.passengerSignalFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'passenger' & handles.data.signalType == 'signal' & ...
    handles.data.sampleID == handles.sampleID, { 'fileName' } ) );

set( handles.displayPassengerSignalFile, 'String', [ handles.passengerSignalPathName{ 1 } handles.passengerSignalFileName{ 1 } ] );
set( handles.displayPassengerSignalFile, 'Enable', 'on' );

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

[ handles.passengerSignal_Amp, handles.passengerSFs ] = wavread( [ handles.passengerSignalPathName{ 1 } handles.passengerSignalFileName{ 1 } ] );
handles.passengerSignal_Amp = handles.passengerSignal_Amp( :, passengerChannel );

```

```
set( handles.controlPassengerSignalChannel, 'Value', passengerChannel );
set( handles.controlPassengerSignalChannel, 'Enable', 'off' );

%-------------------------------------------------------------------------
% 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' );

%-------------------------------------------------------------------------
% Front Signal
%-------------------------------------------------------------------------
if strcmp( opCond, 'Stationary' ) == 1
    set( handles.textFront, 'Enable', 'on' );
    set( handles.textFrSig, 'Enable', 'on' );
    set( handles.textFrPreAmb, 'Enable', 'on' );
    handles.frontimeTriggerSignalPathName = cellstr( handles.data( handles.data.opCond == opCond &
    handles.data.micSide == 'front' & handles.data.signalType == 'signal' & ...
        handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
    handles.frontimeTriggerSignalFileName = cellstr( handles.data( handles.data.opCond == opCond &
    handles.data.micSide == 'front' & handles.data.signalType == 'signal' & ...
        handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
    set( handles.displayFrontMicSignalFile, 'String', [ handles.frontimeTriggerSignalPathName{ 1 }
    handles.frontimeTriggerSignalFileName{ 1 } ] );
    set( handles.displayFrontMicSignalFile, 'Enable', 'on' );

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

    [ handles.frontSignal_Amp, handles.frontSFs ] = wavread( [ handles.frontimeTriggerSignalPathName{ 1 }
    handles.frontimeTriggerSignalFileName{ 1 } ] );
    handles.frontSignal_Amp = handles.frontSignal_Amp( :, frontChannel );

    set( handles.controlFrontMicSignalChannel, 'Value', frontChannel );
    set( handles.controlFrontMicSignalChannel, 'Enable', 'off' );

    %-----------------------------------------------------------------------
    % Front Pre-ambient
    %-----------------------------------------------------------------------
    handles.frontPreAmbientPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
    'front' & handles.data.signalType == 'preAmbient' & ...
        handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
    handles.frontPreAmbientFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide ==
    'front' & handles.data.signalType == 'preAmbient' & ...
        handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
    set( handles.displayFrontMicPreAmbientFile, 'String', [ handles.frontPreAmbientPathName{ 1 }
    handles.frontPreAmbientFileName{ 1 } ] );
    set( handles.displayFrontMicPreAmbientFile, 'Enable', 'on' );

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

    [ handles.frontPreAmbient_Amp, handles.frontAFs ] = wavread( [ handles.frontPreAmbientPathName{ 1 }
    handles.frontPreAmbientFileName{ 1 } ] );
    handles.frontPreAmbient_Amp = handles.frontPreAmbient_Amp( :, frontPreAmbientChannel );
```

```
    set( handles.controlFrontMicPreAmbientChannel, 'Value', frontPreAmbientChannel );
    set( handles.controlFrontMicPreAmbientChannel, 'Enable', 'off' );

    %-------------------------------------------------------------------------
    % Front Post-ambient
    %-------------------------------------------------------------------------
    handles.frontPostAmbientPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'front' & handles.data.signalType == 'postAmbient' & ...
        handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
    handles.frontPostAmbientFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.micSide == 'front' & handles.data.signalType == 'postAmbient' & ...
        handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
    set( handles.displayFrontMicPostAmbientFile, 'String', [ handles.frontPostAmbientPathName{ 1 } handles.frontPostAmbientFileName{ 1 } ] );
    set( handles.displayFrontMicPostAmbientFile, 'Enable', 'on' );

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

    [ handles.frontPostAmbient_Amp, handles.frontAFs ] = wavread( [ handles.frontPostAmbientPathName{ 1 } handles.frontPostAmbientFileName{ 1 } ] );
    handles.frontPostAmbient_Amp = handles.frontPostAmbient_Amp( :, frontPostAmbientChannel );

    set( handles.controlFrontMicPostAmbientChannel, 'Value', frontPostAmbientChannel );
    set( handles.controlFrontMicPostAmbientChannel, 'Enable', 'off' );

end

%-------------------------------------------------------------------------
% Trigger
%-------------------------------------------------------------------------
if ~ isempty( handles.data( handles.data.opCond == opCond & handles.data.signalType == 'trigger' & handles.data.sampleID == handles.sampleID, : ) )
    handles.triggerPathName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.signalType == 'trigger' & handles.data.sampleID == handles.sampleID, { 'pathName' } ) );
    handles.triggerFileName = cellstr( handles.data( handles.data.opCond == opCond & handles.data.signalType == 'trigger' & handles.data.sampleID == handles.sampleID, { 'fileName' } ) );
    set( handles.displayTriggerFile, 'String', [ handles.triggerPathName{ 1 } handles.triggerFileName{ 1 } ] );
    set( handles.displayTriggerFile, 'Enable', 'on' );

    triggerChannel = double( handles.data( handles.data.opCond == opCond & handles.data.signalType == 'trigger' & ...
        handles.data.sampleID == handles.sampleID, { 'channel' } ) );

    [ handles.triggerSignal_Amp, handles.triggerFs ] = wavread( [ handles.triggerPathName{ 1 } handles.triggerFileName{ 1 } ] );
    handles.triggerSignal_Amp = handles.triggerSignal_Amp( :, triggerChannel );

    set( handles.controlTriggerChannel, 'Value', triggerChannel);
```

```
    set( handles.controlTriggerChannel, 'Enable', 'off' );


    tThresh1 = handles.data( handles.data.opCond == opCond & handles.data.signalType == 'trigger' & handles.data.sampleID == handles.sampleID, { 'TriggerThreshold' } );
    tThresh2 = tThresh1.TriggerThreshold ;
    tThresh3 = cell2mat( tThresh2 ) ;

    set( handles.editTriggerThreshold, 'String', tThresh3 );

    set( handles.checkboxTrigger,      'Value', 1 );
    set( handles.checkboxTrigger,      'Enable', 'on' );
    set( handles.editTriggerThreshold, 'Enable', 'on' );
end

function handles = plotData( handles )

% Create time signals for all signals
handles.driverSignal_Time = ( 0 : length( handles.driverSignal_Amp ) - 1 )/handles.driverSFs;%SFs = signal, Fs
handles.driverPreAmbient_Time = ( 0 : length( handles.driverPreAmbient_Amp ) - 1 )/handles.driverAFs;% AFs = ambient, Fs
handles.driverPostAmbient_Time = ( 0 : length( handles.driverPostAmbient_Amp ) - 1 )/handles.driverAFs;
handles.passengerSignal_Time = ( 0 : length( handles.passengerSignal_Amp ) - 1 )/handles.passengerSFs;
handles.passengerPreAmbient_Time = ( 0 : length( handles.passengerPreAmbient_Amp ) -1 )/handles.passengerAFs;
handles.passengerPostAmbient_Time = ( 0 : length( handles.passengerPostAmbient_Amp ) -1 )/handles.passengerAFs;

if strcmp( handles.opCond, 'Stationary' ) == 1
    handles.frontSignal_Time = ( 0 : length( handles.frontSignal_Amp ) - 1 )/handles.frontSFs;
    handles.frontPreAmbient_Time = ( 0 : length( handles.frontPreAmbient_Amp ) -1 )/handles.frontAFs;
    handles.frontPostAmbient_Time = ( 0 : length( handles.frontPostAmbient_Amp ) -1 )/handles.frontAFs;

end

if strcmp (handles.opCond,'10 km/h' ) || strcmp (handles.opCond,'20 km/h' )||  strcmp (handles.opCond,'30 km/h' );
    if get( handles.checkboxTrigger, 'Value' ) == 1
        handles.triggerSignal_Time = ( 0 : length( handles.triggerSignal_Amp ) - 1 )/handles.triggerFs;
    else % If trigger doesn't exist, create one that uses the whole signal --> trigger checkbox is always checked, so this shouldn't be needed.
        handles.triggerSignal_Amp =  0 * handles.driverSignal_Time;
        handles.triggerSignal_Amp( end ) = 1;
        handles.triggerSignal_Time = handles.driverSignal_Time;
    end
end

cla( handles.axesDisplay )
axes( handles.axesDisplay ) % Need to explicitly set this or it can change

% Clear old signals
if isfield( handles, 'ax' )
    if length ( handles.ax ) > 1
```

```
        set( handles.ax( 2 ), 'Visible' , 'Off' )
        cla( handles.ax( 2 ) )
    end
else
    handles.ax = gca ;
end

% Plot new signals
if isfield( handles, 'driverSignal_Amp' )

    handles.hPlotimeTriggerSignal_driver = ...
        plot( handles.axesDisplay, handles.driverSignal_Time, handles.driverSignal_Amp, 'color', [0,0,1] );

    % Yaxis label
    set( get( handles.ax( 1 ), 'ylabel' ), 'String', 'Acoustic Pressure (Pa)' ) ;

    pause( 1 ) ;

    if isfield ( handles, 'hAxis' )
        set(handles.hAxis(2), 'Visible', 'off') ;
    end

    hold on

    handles.hPlotPreAmbient_driver = ...
        plot( handles.axesDisplay, handles.driverPreAmbient_Time, handles.driverPreAmbient_Amp, 'color', [0,0.6,1] );
    pause( 1 ) ;

    handles.hPlotPostAmbient_driver = ...
        plot( handles.axesDisplay, handles.driverPostAmbient_Time, handles.driverPostAmbient_Amp, 'color', [0,1,1] );
    pause( 1 ) ;

end

if isfield( handles, 'passengerSignal_Amp' )
    handles.hPlotimeTriggerSignal_passenger = ...
        plot( handles.axesDisplay, handles.passengerSignal_Time, handles.passengerSignal_Amp, 'color', [1,0,0] );
    pause( 1 ) ;

    handles.hPlotPreAmbient_passenger = ...
        plot( handles.axesDisplay, handles.passengerPreAmbient_Time, handles.passengerPreAmbient_Amp, 'color', [1,0,0.4]  );
    pause( 1 ) ;

    handles.hPlotPostAmbient_passenger = ...
        plot( handles.axesDisplay, handles.passengerPostAmbient_Time, handles.passengerPostAmbient_Amp, 'color', [1,0,1]  );
    pause( 1 ) ;

```

```
end

if strcmp ( handles.opCond,'Stationary' )
    if isfield( handles, 'frontSignal_Amp' )
        handles.hPlotimeTriggerSignal_front = ...
            plot( handles.axesDisplay, handles.frontSignal_Time, handles.frontSignal_Amp, 'color', [0,0,0] );
        pause( 1 ) ;

        handles.hPlotPreAmbient_front = ...
            plot( handles.axesDisplay, handles.frontPreAmbient_Time, handles.frontPreAmbient_Amp, 'color',
            [0.3,0.3,0.3]);
        pause( 1 ) ;

        handles.hPlotPostAmbient_front = ...
            plot( handles.axesDisplay, handles.frontPostAmbient_Time, handles.frontPostAmbient_Amp, 'color',
            [0.6,0.6,0.6] );
        pause( 1 ) ;
    end
end

if strcmp (handles.opCond,'10 km/h' ) || strcmp (handles.opCond,'20 km/h' )||  strcmp (handles.opCond,'30 km/h' );
    if get( handles.checkboxTrigger, 'Value' ) == 1
        % Prep trigger percentage plot
        [ handles.MaxTriggerSignal, handles.MaxTriggerSignalIdx ] = max( abs( handles.triggerSignal_Amp ) );
        handles.timeAtMaxTrigger = handles.triggerSignal_Time( handles.MaxTriggerSignalIdx );
        handles.triggerScaledTo100 = ( handles.triggerSignal_Amp/handles.MaxTriggerSignal ) * 100;
        handles.triggerThresholdValue = str2num( get( handles.editTriggerThreshold, 'String' ) );
        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 )

        % Plot trigger signal
        if length ( handles.ax ) > 1
            set( handles.ax( 2 ), 'Visible' , 'On' )
            axes( handles.ax( 2 ) ) ;
            hold on
            handles.hPlotTrigger = plot( handles.ax( 2 ), handles.triggerSignal_Time, handles.triggerScaledTo100, 'g'
            ,'LineWidth', 2);

        else
            ax1hv = get( handles.ax( 1 ), 'HandleVisibility' ) ;
            handles.ax( 2 ) = axes( 'HandleVisibility', ax1hv, 'Units', get( handles.ax( 1 ), 'Units' ), ...
                'Position',get( handles.ax( 1 ), 'Position' ), 'Parent', get( handles.ax( 1 ),'Parent' ) );

            handles.hPlotTrigger = plot( handles.ax( 2 ), handles.triggerSignal_Time, handles.triggerScaledTo100, 'g'
            ,'LineWidth', 2);

            set( handles.ax( 2 ),'YAxisLocation','right','Color','none', ...
```

```
                'XGrid','off','YGrid','off','Box','off', ...
                'HitTest','off');
            xlim2 = get( handles.ax( 2 ),'XLim' );
            ylim2 = get( handles.ax( 2 ), 'YLim' );

            set( handles.ax( 2 ), 'Xtick', [] )
            set( handles.ax( 2 ), 'YLim', [0 100 ] )
            set( handles.ax( 2 ), 'Ytick', [0:10:100] )
            set( handles.ax( 2 ), 'YTickLabel', [0:10:100] )

            % Fix extra tick marks on 2nd y axis
            linkaxes( handles.ax, 'x' );
            set( handles.ax( 1 ), 'Box', 'off' )
            set( handles.ax( 2 ), 'Box', 'off' )
            saveXTick = get( handles.ax( 1 ), 'XTick' ) ;
            set( handles.ax( 2 ), 'XTick', saveXTick, 'XTickLabel', '', 'XAxisLocation', 'Top' )

        end

        % Plot trigger threshold line
        handles.trigX1 = min( handles.triggerSignal_Time );
        handles.trigX2 = max( handles.triggerSignal_Time );

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

    end
end

if strcmp (handles.opCond,'10 km/h' ) || strcmp (handles.opCond,'20 km/h' )||  strcmp (handles.opCond,'30 km/h' );
    set( get( handles.ax( 2 ), 'ylabel' ), 'String', 'Percent of Max Trigger Level (%)' ) ;
else
    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 )]) ;
    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 )]) ;
    axis( [ minSignal maxSignal -inf inf ] )
end

% xlabel( 'Time, seconds' )
set( get( handles.ax( 1 ), 'ylabel' ), 'String', 'Acoustic Pressure (Pa)' ) ;

% title('Determine Analysis Region')
set( handles.axesDisplay, 'XScale', 'linear' )
```

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

% Clear data
handles.data = [] ;
handles.samplesDone = [] ;
handles.numSamples = [] ;
handles.sampleID = [] ;
handles.driverSignalPathName = [] ;
handles.driverSignalFileName = [] ;
handles.driverSignal_Amp = [] ;
handles.driverPreAmbientPathName = [] ;
handles.driverPreAmbientFileName = [] ;
handles.driverPreAmbient_Amp = [] ;
handles.driverPostAmbientPathName = [] ;
handles.driverPostAmbientFileName = [] ;
handles.driverPostAmbient_Amp = [] ;
handles.passengerSignalPathName = [] ;
handles.passengerSignalFileName = [] ;
handles.passengerSignal_Amp = [] ;
handles.passengerPreAmbientPathName = [] ;
handles.passengerPreAmbientFileName = [] ;
handles.passengerPreAmbient_Amp = [] ;
handles.passengerPostAmbientPathName = [] ;
handles.passengerPostAmbientFileName = [] ;
handles.passengerPostAmbient_Amp = [] ;
handles.triggerPathName = [] ;
handles.triggerFileName = [] ;
handles.triggerSignal_Amp = [] ;
handles.timeAtMaxTrigger = [] ;
handles.triggerScaledTo100 = [] ;
handles.triggerThresholdValue = [] ;
handles.idxAtTriggerThreshold = [] ;
handles.timeAtTriggerThreshold = [] ;
handles.MaxTriggerSignal = [] ;
handles.MaxTriggerSignalIdx = [] ;
handles.hPlotimeTriggerSignal_driver = [] ;
handles.hPlotPreAmbient_driver = [] ;
handles.hPlotPostAmbient_driver = [] ;
handles.hPlotimeTriggerSignal_passenger = [] ;
handles.hPlotPreAmbient_passenger = [] ;
handles.hPlotPostAmbient_passenger = [] ;

guidata( hObject, handles ) ; % Update handles

% Clear Displays (Vehicle Information, User Comments)
set( handles.editYear,                        'String', 'Edit Text') ;
```

```
set( handles.editMake,                        'String', 'Edit Text') ;
set( handles.editModel,                       'String', 'Edit Text') ;
set( handles.editNHTSANumber,                 'String', 'Edit Text') ;
set( handles.editVIN,                         'String', 'Edit Text') ;
set( handles.editUserComments,                'String', 'Edit Text') ;

% Clear Displays (Data Review)
set( handles.editSampleID,                      'String', ' NA') ;
set( handles.textTotalSamplesafterDiscard,      'String', ' NA') ;
set( handles.textCurrentSampleNumber,           'String', ' NA') ;
set( handles.textTotalSampleNumbers,            'String', ' NA') ;

set( handles.controlOperatingCondition,         'Value', 1 ) ;
set( handles.displayDriverSignalFile,           'String', ' NA') ;
set( handles.controlDriverSignalChannel,        'Value', 1 );
set( handles.displayDriverPreAmbientFile,       'String', ' NA') ;
set( handles.controlDriverPreAmbientChannel,    'Value', 1 );
set( handles.displayDriverPostAmbientFile,      'String', ' NA') ;
set( handles.controlDriverPostAmbientChannel,   'Value', 1 );
set( handles.displayPassengerSignalFile,        'String', ' NA') ;
set( handles.controlPassengerSignalChannel,     'Value', 2 );
set( handles.displayPassengerPreAmbientFile,    'String', ' NA') ;
set( handles.controlPassengerPreAmbientChannel, 'Value', 2 );
set( handles.displayPassengerPostAmbientFile,   'String', ' NA') ;
set( handles.controlPassengerPostAmbientChannel,'Value', 2 );
set( handles.displayFrontMicSignalFile,         'String', ' NA') ;
set( handles.controlFrontMicSignalChannel,      'Value', 3 );
set( handles.displayFrontMicPreAmbientFile,     'String', ' NA') ;
set( handles.controlFrontMicPreAmbientChannel,  'Value', 3 );
set( handles.displayFrontMicPostAmbientFile ,   'String', ' NA') ;
set( handles.controlFrontMicPostAmbientChannel, 'Value', 3 );
set( handles.displayTriggerFile,                'String', ' NA') ;
set( handles.controlTriggerChannel,             'Value', 3 );
set( handles.editTrigger,                       'String', ' Edit Text') ;

if isfield( handles, 'ax' )
    if length ( handles.ax ) > 1
        set( handles.ax( 2 ), 'Visible' , 'Off' )
        cla( handles.ax( 2 ) )
    end
end

cla( handles.axesDisplay )
axes( handles.axesDisplay )

% Clear Displays & Adjust Visibility (Compute and Resutls)
set( handles.displayRunDurationValue,     'String', '-' );
set( handles.textProcessingStep,          'String', 'Processing sample' );
set( handles.displayRunStatusCurrentFile, 'String', '0' );
set( handles.displayRunStatusTotalFiles,  'String', '0' );
```

```
set( handles.displayCurrentOC,            'String', ' NA' );

set( handles.display2dBError,             'Visible', 'off' );
set( handles.displayAnalysisComplete,     'Visible', 'off' );
set( handles.textAnalysisStopped,         'Visible', 'off' );
set( handles.textRunStatusOf,             'Visible', 'on' );
set( handles.displayRunStatusCurrentFile, 'Visible', 'on' );
set( handles.displayRunStatusTotalFiles,  'Visible', 'on' );
set( handles.displayCurrentOC,            'Visible', 'on' );
set( handles.textRunStatusFor,            'Visible', 'on' );

set( handles.displayOutputFilePath,       'String', ' NA'  );
set( handles.uitableResults,              'Data', [] )
set( handles.displayCompliance,           'String', ' NA') ;

% Clear Saved Data For Tables In GUI_fileLoad2
setappdata( 0, 'loadedTableDataFlag', 0 ) ;
setappdata( 0, 'loadedTableDataStationary', [] ) ;
setappdata( 0, 'loadedTableData10kmh', [] ) ;
setappdata( 0, 'loadedTableData20kmh', [] ) ;
setappdata( 0, 'loadedTableData30kmh', [] ) ;
setappdata( 0, 'loadedTableDataReverse', [] ) ;



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

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

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

% --- Executes during object creation, after setting all properties.
function controlFileSelect_CreateFcn(hObject, eventdata, handles)
% hObject    handle to controlFileSelect (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 editYear_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editYear (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 editMake_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editMake (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 editModel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editModel (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 editNHTSANumber_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editNHTSANumber (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 editVIN_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editVIN (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 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

% --- 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 displayFrontMicSignalFile_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 controlFrontMicSignalChannel_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 displayFrontMicPreAmbientFile_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 controlFrontMicPreAmbientChannel_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 displayFrontMicPostAmbientFile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to displayFrontMicPostAmbientFile (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 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'))
    set(hObject,'BackgroundColor',[0.941, 0.941, 0.941]);
end


%*************************************************************************
%*************************************************************************
%      Empty Callbacks / Not Currently In Use
%*************************************************************************
%*************************************************************************

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

function displayTriggerFile_Callback(hObject, eventdata, handles)
% hObject    handle to displayTriggerFile (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 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

```