```matlab

function varargout = GUI_fileLoad2(varargin)
% GUI_fileLoad2 MATLAB code for GUI_fileLoad2.fig
%      GUI_fileLoad2, by itself, creates a new GUI_fileLoad2 or raises the existing
%      singleton*.
%
%      H = GUI_fileLoad2 returns the handle to a new GUI_fileLoad2 or the handle to
%      the existing singleton*.
%
%      GUI_fileLoad2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_fileLoad2.M with the given input arguments.
%
%      GUI_fileLoad2('Property','Value',...) creates a new GUI_fileLoad2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_fileLoad2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_fileLoad2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_fileLoad2

% Last Modified by GUIDE v2.5 01-Aug-2018 13:22:55

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

%-------------------------------------------------------------------------
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @GUI_fileLoad2_OpeningFcn, ...
    'gui_OutputFcn',  @GUI_fileLoad2_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_fileLoad2 is made visible.
function GUI_fileLoad2_OpeningFcn(hObject, ~, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_fileLoad2 (see VARARGIN)

% Clear the console display
clc

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

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

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

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

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

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

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

```
if handles.loadedTableDataFlag % Load data into the tables if data exists
    if handles.useTestData
        if ~isempty( handles.data )
            for ink = 1 : length( opC )
                handles = displayTable( handles, opC{ ink }, tab{ ink } );
            end
        end
    else
        handles.loadedTableDataStationary = getappdata( 0, 'loadedTableDataStationary' );
        handles.loadedTableData10kmh = getappdata( 0, 'loadedTableData10kmh' );
        handles.loadedTableData20kmh = getappdata( 0, 'loadedTableData20kmh' );
        handles.loadedTableData30kmh = getappdata( 0, 'loadedTableData30kmh' );
        handles.loadedTableDataReverse = getappdata( 0, 'loadedTableDataReverse' );

        set( handles.tableStationary, 'Data', handles.loadedTableDataStationary );
        set( handles.table10kmh, 'Data', handles.loadedTableData10kmh );
        set( handles.table20kmh, 'Data', handles.loadedTableData20kmh );
        set( handles.table30kmh, 'Data', handles.loadedTableData30kmh );
        set( handles.tableReverse, 'Data', handles.loadedTableDataReverse );
    end

    % Enable the submit data button if at least for samples for each operating condition
    if size (handles.loadedTableDataStationary, 1 ) >= 4 && size (handles.loadedTableData10kmh , 1 ) >= 4 && size
    (handles.loadedTableData20kmh , 1 ) >= 4 ...
            && size (handles.loadedTableData30kmh  , 1 ) >= 4 && size (handles.loadedTableDataReverse , 1 ) >= 4
        set( handles.buttonSubmitData, 'Enable', 'on' );
        set( handles.textDataSubmittalMessage, 'String', ' ' );
    else
        set( handles.buttonSubmitData, 'Enable', 'off' );
    end
end

if isfield( handles, 'fileLoadCondition' )
    if strcmp( handles.fileLoadCondition, 'batch' )
        handles = displayTable( handles ); % Display batch loaded data in tables if importing by batch
    end
end

% Update handles structure
guidata(hObject, handles);

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

% --- Executes on selection change in OperatingConditionFileLoadMenu.
function OperatingConditionFileLoadMenu_Callback(hObject, eventdata, handles)
```

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

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

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

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

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

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

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

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

```
        set(handles.textTriggerTitle, 'enable', 'on')
        set(handles.textTrigger, 'enable', 'on')
        set(handles.pushbuttonSelectTrigger, 'enable', 'on')
        set(handles.controlTriggerChannel, 'enable', 'on')

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

    case '30 km/h'
        handles.opCond = '30 km/h';
        set(handles.textTriggerTitle, 'enable', 'on')
        set(handles.textTrigger, 'enable', 'on')
        set(handles.pushbuttonSelectTrigger, 'enable', 'on')
        set(handles.controlTriggerChannel, 'enable', 'on')

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

    case 'Reverse'
        handles.opCond = 'Reverse';
        set(handles.textTriggerTitle, 'enable', 'off')
        set(handles.textTrigger, 'enable', 'off')
        set(handles.pushbuttonSelectTrigger, 'enable', 'off')
        set(handles.controlTriggerChannel, 'enable', 'off')

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

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

% Update handles structure
guidata( hObject, handles );

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

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

guidata( hObject, handles );

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

getFileFilter = '*.wav';

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

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

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

% Update handles structure
guidata( hObject, handles );

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

% Update handles structure
guidata( hObject, handles );

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

getFileFilter = '*.wav';

```

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

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

if handles.driverPreAmbientFileName ~= 0
    set( handles.displayDriverPreAmbientFile, 'String', [ handles.driverPreAmbientPathName, '\',
    handles.driverPreAmbientFileName ] );
    set( handles.displayDriverPreAmbientFile, 'Enable', 'on' );
end

% Update handles structure
guidata( hObject, handles );

% --- Executes on selection change in controlDriverPreAmbientChannel.
function controlDriverPreAmbientChannel_Callback(hObject, ~, handles)
handles.driverPreAmbientChannel = get( hObject, 'Value' );

% Update handles structure
guidata( hObject, handles );

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

getFileFilter = '*.wav';

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

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

if handles.driverPostAmbientFileName ~= 0
    set( handles.displayDriverPostAmbientFile, 'String', [ handles.driverPostAmbientPathName, '\',
    handles.driverPostAmbientFileName ] );
    set( handles.displayDriverPostAmbientFile, 'Enable', 'on' );
end

% Update handles structure
guidata( hObject, handles );

% --- Executes on selection change in controlDriverPostAmbientChannel.
function controlDriverPostAmbientChannel_Callback(hObject, eventdata, handles)
handles.driverPostAmbientChannel = get( hObject, 'Value' );
```

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

% --- Executes on button press in pushbuttonSelectPassengerSignal.
function pushbuttonSelectPassengerSignal_Callback(hObject, ~, handles)

getFileFilter = '*.wav';

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

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

if handles.passengerSignalFileName ~= 0
    set( handles.displayPassengerSignalFile, 'String', [ handles.passengerSignalPathName, '\',
    handles.passengerSignalFileName ] );
    set( handles.displayPassengerSignalFile, 'Enable', 'on' );
end

% Update handles structure
guidata( hObject, handles );

% --- Executes on selection change in controlPassengerSignalChannel.
function controlPassengerSignalChannel_Callback(hObject, ~, handles)
handles.passengerChannel = get( hObject, 'Value' );

% Update handles structure
guidata( hObject, handles );

% --- Executes on button press in pushbuttonSelectPassengerPreAmbient.
function pushbuttonSelectPassengerPreAmbient_Callback(hObject, ~, handles)

getFileFilter = '*.wav';

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

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

if handles.passengerPreAmbientFileName ~= 0
    set( handles.displayPassengerPreAmbientFile, 'String', ...
        [ handles.passengerPreAmbientPathName, '\', handles.passengerPreAmbientFileName ] );
    set( handles.displayPassengerPreAmbientFile, 'Enable', 'on' );
end
```

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

% --- Executes on selection change in controlPassengerPreAmbientChannel.
function controlPassengerPreAmbientChannel_Callback(hObject, ~, handles)
handles.passengerPreAmbientChannel = get( hObject, 'Value' );

% Update handles structure
guidata( hObject, handles );

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

getFileFilter = '*.wav';

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

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

if handles.passengerPostAmbientFileName ~= 0
    set( handles.displayPassengerPostAmbientFile, 'String', [ handles.passengerPostAmbientPathName, '\',
    handles.passengerPostAmbientFileName ] );
    set( handles.displayPassengerPostAmbientFile, 'Enable', 'on' );
end

% Update handles structure
guidata( hObject, handles );

% --- Executes on selection change in controlPassengerPostAmbientChannel.
function controlPassengerPostAmbientChannel_Callback(hObject, eventdata, handles)
handles.passengerPostAmbientChannel = get( hObject, 'Value' );

% Update handles structure
guidata( hObject, handles );

% --- Executes on button press in pushbuttonSelectFrontSignal.
function pushbuttonSelectFrontSignal_Callback(hObject, ~, handles)

getFileFilter = '*.wav';

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

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

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

% Update handles structure
guidata( hObject, handles );

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

% Update handles structure
guidata( hObject, handles );

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

getFileFilter = '*.wav';

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

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

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

% Update handles structure
guidata( hObject, handles );

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

% Update handles structure
guidata( hObject, handles );

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

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

getFileFilter = '*.wav';

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

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

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

% Update handles structure
guidata( hObject, handles );

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

% Update handles structure
guidata( hObject, handles );

function pushbuttonSelectTrigger_Callback(hObject, ~, handles)

getFileFilter = '*.wav';

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

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

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

% Update handles structure
guidata(hObject, handles);

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

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

% --- Executes on button press in pushbuttonImportData.
function pushbuttonImportData_Callback(hObject, ~, handles)

set( handles.displayDataImportMessages, 'String', '');

% Display error if operating condition not selected
if ~ isfield(handles, 'operatingCondition') ;
    set( handles.displayDataImportMessages, 'String', 'Select Operating Condition', 'ForegroundColor',[0.847,0.161,0]
    );
end

% Get info from GUI
if ~ isfield( handles, 'opCond' )
    contents = cellstr( get( handles.controlOperatingCondition, 'String' ) );
    handles.opCond = contents{ get( handles.controlOperatingCondition, 'Value' ) };
end

if ~ isfield( handles, 'sampleID' )
    handles.sampleID = get( handles.editSampleID, 'String' );
end

if ~ isfield( handles, 'driverChannel' )
    handles.driverChannel = get( handles.controlDriverSignalChannel, 'Value' );
end

if ~ isfield( handles, 'driverPreAmbientChannel' )
    handles.driverPreAmbientChannel = get( handles.controlDriverPreAmbientChannel, 'Value' );
end

if ~ isfield( handles, 'driverPostAmbientChannel' )
    handles.driverPostAmbientChannel = get( handles.controlDriverPostAmbientChannel, 'Value' );
end

if ~ isfield( handles, 'passengerChannel' )
    handles.passengerChannel = get( handles.controlPassengerSignalChannel, 'Value' );
end

if ~ isfield( handles, 'passengerPreAmbientChannel' )
    handles.passengerPreAmbientChannel = get( handles.controlPassengerPreAmbientChannel, 'Value' );
end

if ~ isfield( handles, 'passengerPostAmbientChannel' )
    handles.passengerPostAmbientChannel = get( handles.controlPassengerPostAmbientChannel, 'Value' );
end

if ~ isfield( handles, 'frontChannel' )
    handles.frontChannel = get( handles.controlFrontSignalChannel, 'Value' );
```

```
end

if ~ isfield( handles, 'frontPreAmbientChannel' )
    handles.frontPreAmbientChannel = get( handles.controlFrontPreAmbientChannel, 'Value' );
end

if ~ isfield( handles, 'frontPostAmbientChannel' )
    handles.frontPostAmbientChannel = get( handles.controlFrontPostAmbientChannel, 'Value' );
end

if ~ isfield( handles, 'triggerChannel' )
    handles.triggerChannel = get( handles.controlTriggerChannel, 'Value' );
end

%Set up variables
switch handles.opCond
    case 'Stationary'
        opCond = nominal( ' Stationary' );
        table = 'tableStationary';
    case '10 km/h'
        opCond = nominal( ' 10 km/h' );
        table = 'table10kmh';
    case '20 km/h'
        opCond = nominal( ' 20 km/h' );
        table = 'table20kmh';
    case '30 km/h'
        opCond = nominal( ' 30 km/h' );
        table = 'table30kmh';
    case 'Reverse'
        opCond = nominal( ' Reverse' );
        table = 'tableReverse';
end

%Pull in data
ink = getappdata( 0, 'ink' );

if isempty( ink )
    ink = 1;
end

if strcmp( handles.sampleID, 'NA' ) || isempty ( handles.sampleID ) ; %If user does not input sample ID and no data has been entered, use ink to define sample ID
    if isempty( handles.data ) ;
        handles.sampleID = num2str( ink );
    else
        handles.sampleID = num2str( max ( str2num ( char (handles.data.sampleID ) ) )+ 1 ) ; % If data already exists, increment the max sample number
    end
end
```

```
% Checking to make sure all mics have signal and ambient file selected
if isfield( handles, 'driverSignalPathName' ) && isfield( handles, 'driverPreAmbientPathName' ) && isfield( handles, 'driverPostAmbientPathName' ) && isfield( handles, 'passengerSignalPathName' ) ...
        && isfield( handles, 'passengerPreAmbientPathName' )  && isfield( handles, 'passengerPostAmbientPathName' )&& ~isempty( handles.driverSignalPathName ) && ~isempty( handles.driverPreAmbientPathName' ) && ~isempty( handles.driverPostAmbientPathName' ) && ~isempty( handles.passengerSignalPathName ) ...
        && ~isempty( handles.passengerPreAmbientPathName )&& ~isempty( handles.passengerPostAmbientPathName )

    handles = dataImport( handles ); % import into handles.data

    %If stationary OC, import front data
    if strcmp( char(opCond), 'Stationary' )

        %Error messages if field not selected properly
        if ~ isfield( handles, 'frontSignalPathName' ) || isempty ( handles.frontSignalPathName ) ;
            set( handles.displayDataImportMessages, 'String', 'Select signal file for Front Mic', 'ForegroundColor',[0.847,0.161,0]);
        elseif handles.frontSignalPathName == 0
            set( handles.displayDataImportMessages, 'String', 'Select signal file for Front Mic', 'ForegroundColor',[0.847,0.161,0]);

        elseif ~ isfield( handles, 'frontPreAmbientPathName' ) || isempty ( handles.frontPreAmbientPathName ) ;
            set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Front Mic', 'ForegroundColor',[0.847,0.161,0]);
        elseif handles.frontPreAmbientPathName == 0
            set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Front Mic', 'ForegroundColor',[0.847,0.161,0]);

        elseif ~ isfield( handles, 'frontPostAmbientPathName' ) || isempty ( handles.frontPostAmbientPathName ) ;
            set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Front Mic', 'ForegroundColor',[0.847,0.161,0]);
        elseif handles.frontPostAmbientPathName == 0
            set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Front Mic', 'ForegroundColor',[0.847,0.161,0]);

        end

        handles = frontImport( handles ); % import into handles.data
    end

    % if passby, import trigger data
    if strcmp( char(opCond), '10 km/h' ) || strcmp( char(opCond), '20 km/h' )|| strcmp( char(opCond), '30 km/h' )
        if ~ isfield( handles, 'triggerFileName' ) || isempty ( handles.triggerFileName ) ;
            set( handles.displayDataImportMessages, 'String', 'Select trigger file', 'ForegroundColor',[0.847,0.161,0]);
        elseif ( handles.triggerFileName ) == 0
            set( handles.displayDataImportMessages, 'String', 'Select trigger file', 'ForegroundColor',[0.847,0.161,0]);
        end
        handles = triggerImport( handles );  % import into handles.data
```

```
    end

    handles = displayTable( handles, opCond, table ); % display in tables
    handles = resetDisplays( handles ); % reset file selections

    tableStationarySampleList = unique( handles.data( handles.data.opCond == 'Stationary' , { 'sampleID' } ) );
    table10kmhSampleList = unique( handles.data( handles.data.opCond == '10 km/h' , { 'sampleID' } ) );
    table20kmhSampleList = unique( handles.data( handles.data.opCond == '20 km/h' , { 'sampleID' } ) );
    table30kmhSampleList = unique( handles.data( handles.data.opCond == '30 km/h' , { 'sampleID' } ) );
    tableReverseSampleList = unique( handles.data( handles.data.opCond == 'Reverse' , { 'sampleID' } ) );

    if ~handles.useTestData % If not using test data, check to see if there are at least 4 samples for each operating condition. Do not enable Submit Data button until 4 samples for each OC
        if size (tableStationarySampleList, 1 ) >= 4 && size (table10kmhSampleList, 1 ) >= 4 && size (table20kmhSampleList, 1 ) >= 4 ...
                && size (table30kmhSampleList, 1 ) >= 4 && size (tableReverseSampleList, 1 ) >= 4
            set( handles.buttonSubmitData, 'Enable', 'on' );
            set( handles.textDataSubmittalMessage, 'String', ' ' );
        else
            set( handles.buttonSubmitData, 'Enable', 'off' );
            set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to submit data' );
        end
    end

    %Updated handles structure
    guidata( hObject, handles );
    setappdata( 0, 'ink', ink + 1 );
    handles.output = handles.data;

else
    % Write error messages if fields don't exist
    if ~isfield( handles, 'driverSignalPathName' )|| isempty ( handles.driverSignalPathName );
        set( handles.displayDataImportMessages, 'String', 'Select signal file for Driver Side Mic', 'ForegroundColor',[0.847,0.161,0]);

    elseif handles.driverSignalPathName == 0
        set( handles.displayDataImportMessages, 'String', 'Select signal file for Driver Side Mic', 'ForegroundColor',[0.847,0.161,0]);

    elseif ~isfield( handles, 'driverPreAmbientPathName' )|| isempty ( handles.driverPreAmbientPathName );
        set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Driver Side Mic', 'ForegroundColor',[0.847,0.161,0])

    elseif handles.driverPreAmbientPathName == 0
        set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Driver Side Mic', 'ForegroundColor',[0.847,0.161,0])

    elseif ~isfield( handles, 'driverPostAmbientPathName' )|| isempty ( handles.driverPostAmbientPathName );
        set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Driver Side Mic',
```

```
        'ForegroundColor',[0.847,0.161,0])

    elseif handles.driverPostAmbientPathName == 0
        set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Driver Side Mic',
        'ForegroundColor',[0.847,0.161,0])

    elseif ~isfield( handles, 'passengerSignalPathName' )|| isempty ( handles.passengerSignalPathName );
        set( handles.displayDataImportMessages, 'String', 'Select signal file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0])

    elseif handles.passengerSignalPathName == 0
        set( handles.displayDataImportMessages, 'String', 'Select signal file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0])

    elseif ~isfield( handles, 'passengerPreAmbientPathName' )|| isempty ( handles.passengerPreAmbientPathName );
        set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0]);

    elseif handles.passengerPreAmbientPathName == 0
        set( handles.displayDataImportMessages, 'String', 'Select pre-ambient file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0])

    elseif ~isfield( handles, 'passengerPostAmbientPathName' )|| isempty ( handles.passengerPostAmbientPathName );
        set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0]);

    elseif handles.passengerPostAmbientPathName == 0
        set( handles.displayDataImportMessages, 'String', 'Select post-ambient file for Passenger Side Mic',
        'ForegroundColor',[0.847,0.161,0])
    end
end;

% --- Executes on button press in pushbuttonClearStationary.
function pushbuttonClearStationary_Callback(hObject, ~, handles)
% clear data in stationary table and remove from handles.data

set( handles.tableStationary, 'Data', cell( 4, 3 ) );
handles.data( handles.data.opCond == 'Stationary', : ) = [];
handles.datStationary = [];

if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 ) >= 4 ...
        && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
    set( handles.buttonSubmitData, 'Enable', 'on' );
    set( handles.textDataSubmittalMessage, 'String', ' ' );

else
    set( handles.buttonSubmitData, 'Enable', 'off' );
    set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to
    submit data' );
end
```

```
%Update gui handles
guidata( hObject, handles )

% --- Executes on button press in pushbuttonClear10kmh.
function pushbuttonClear10kmh_Callback(hObject, ~, handles)
% clear data in 10kmh table and remove from handles.data

set( handles.table10kmh, 'Data', cell( 4, 3 ) );
handles.data( handles.data.opCond == '10 km/h', : ) = [];
handles.dat10kmh = [];

if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 ) >= 4 ...
        && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
    set( handles.buttonSubmitData, 'Enable', 'on' );
    set( handles.textDataSubmittalMessage, 'String', ' ' );
else
    set( handles.buttonSubmitData, 'Enable', 'off' );
    set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to
    submit data' );
end

%Update gui handles
guidata( hObject, handles )

% --- Executes on button press in pushbuttonClear20kmh.
function pushbuttonClear20kmh_Callback(hObject, ~, handles)
% clear data in 20kmh table and remove from handles.data

set( handles.table20kmh, 'Data', cell( 4, 3 ) );
handles.data( handles.data.opCond == '20 km/h', : ) = [];
handles.dat20kmh = [];

if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 ) >= 4 ...
        && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
    set( handles.buttonSubmitData, 'Enable', 'on' );
    set( handles.textDataSubmittalMessage, 'String', ' ' );
else
    set( handles.buttonSubmitData, 'Enable', 'off' );
    set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to
    submit data' );
end

%Update gui handles
guidata( hObject, handles )

% --- Executes on button press in pushbuttonClear30kmh.
function pushbuttonClear30kmh_Callback(hObject, ~, handles)
% clear data in 30kmh table and remove from handles.data

```

```
set( handles.table30kmh, 'Data', cell( 4, 3 ) );
handles.data( handles.data.opCond == '30 km/h', : ) = [];
handles.dat30kmh = [];

if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 ) >= 4 ...
        && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
    set( handles.buttonSubmitData, 'Enable', 'on' );
    set( handles.textDataSubmittalMessage, 'String', ' ' );
else
    set( handles.buttonSubmitData, 'Enable', 'off' );
    set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to
    submit data' );
end

%Update gui handles
guidata( hObject, handles )

% --- Executes on button press in pushbuttonClearReverse.
function pushbuttonClearReverse_Callback(hObject, ~, handles)
% clear data in reverse table and remove from handles.data

set( handles.tableReverse, 'Data', cell( 4, 3 ) );
handles.data( handles.data.opCond == 'Reverse', : ) = [];

%Update gui handles
guidata( hObject, handles )

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

% ***********FOR TESTING PURPOSES ONLY************
if handles.useTestData
    load('TestData.mat')
    setappdata( 0, 'data', handles.data )

    handles.loadedTableDataFlag = 1 ;
    setappdata( 0, 'loadedTableDataFlag', handles.loadedTableDataFlag ) ;
% ***********For actual application (not testing)************
else
    % Save current table data for persistence between GUI windows
    handles.loadedTableDataFlag = 1 ;
    handles.loadedTableDataStationary = get(handles.tableStationary, 'data') ;
    handles.loadedTableData10kmh = get(handles.table10kmh, 'data') ;
    handles.loadedTableData20kmh = get(handles.table20kmh, 'data') ;
    handles.loadedTableData30kmh = get(handles.table30kmh, 'data') ;
    handles.loadedTableDataReverse = get(handles.tableReverse, 'data') ;

```

```
    setappdata( 0, 'loadedTableDataFlag', handles.loadedTableDataFlag ) ;
    setappdata( 0, 'loadedTableDataStationary', handles.loadedTableDataStationary ) ;
    setappdata( 0, 'loadedTableData10kmh', handles.loadedTableData10kmh ) ;
    setappdata( 0, 'loadedTableData20kmh', handles.loadedTableData20kmh ) ;
    setappdata( 0, 'loadedTableData30kmh', handles.loadedTableData30kmh ) ;
    setappdata( 0, 'loadedTableDataReverse', handles.loadedTableDataReverse ) ;

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

    %Reset displays
    resetDisplays( handles );
end

close
uistack( GUI_mainFinalRule_Option1, 'top' );

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

resetDisplays( handles );
close

uistack( GUI_mainFinalRule_Option1, 'top' );

% --- Outputs from this function are returned to the command line.
function varargout = GUI_fileLoad2_OutputFcn(~, ~, handles)

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

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

function handles = resetDisplays( handles )
% Reset File / Path Names
handles.driverSignalPathName = [];
handles.driverSignalFileName = [];

set( handles.displayDriverSignalFile, 'String', 'NA' );
set( handles.displayDriverSignalFile, 'Enable', 'off' );

handles.driverPreAmbientPathName = [];
handles.driverPreAmbientFileName = [];

handles.driverPostAmbientPathName = [];
handles.driverPostAmbientFileName = [];
```

```
set( handles.displayDriverPreAmbientFile, 'String', 'NA' );
set( handles.displayDriverPreAmbientFile, 'Enable', 'off' );

set( handles.displayDriverPostAmbientFile, 'String', 'NA' );
set( handles.displayDriverPostAmbientFile, 'Enable', 'off' );

handles.passengerSignalPathName = [];
handles.passengerSignalFileName = [];

set( handles.displayPassengerSignalFile, 'String', 'NA' );
set( handles.displayPassengerSignalFile, 'Enable', 'off' );

handles.passengerPreAmbientPathName = [];
handles.passengerPreAmbientFileName = [];

handles.passengerPostAmbientPathName = [];
handles.passengerPostAmbientFileName = [];

set( handles.displayPassengerPreAmbientFile, 'String', 'NA' );
set( handles.displayPassengerPreAmbientFile, 'Enable', 'off' );

set( handles.displayPassengerPostAmbientFile, 'String', 'NA' );
set( handles.displayPassengerPostAmbientFile, 'Enable', 'off' );

handles.frontSignalPathName = [];
handles.frontSignalFileName = [];

set( handles.displayFrontSignalFile, 'String', 'NA' );
set( handles.displayFrontSignalFile, 'Enable', 'off' );

handles.frontPreAmbientPathName = [];
handles.frontPreAmbientFileName = [];

handles.frontPostAmbientPathName = [];
handles.frontPostAmbientFileName = [];

set( handles.displayFrontPreAmbientFile, 'String', 'NA' );
set( handles.displayFrontPreAmbientFile, 'Enable', 'off' );

set( handles.displayFrontPostAmbientFile, 'String', 'NA' );
set( handles.displayFrontPostAmbientFile, 'Enable', 'off' );

handles.triggerPathName = [];
handles.triggerFileName = [];

set( handles.displayTriggerFile, 'String', 'NA' );
set( handles.displayTriggerFile, 'Enable', 'off' );

handles.sampleID = [];
set( handles.editSampleID, 'String', 'NA' );
```

```
function handles = dataImport( handles )
% Import data into handles.data

opCond = handles.opCond;

dataImport = dataset( { { opCond; opCond; opCond; opCond; opCond; opCond }, 'opCond' }, ...
    { { 'driver'; 'driver'; 'driver'; 'passenger'; 'passenger'; 'passenger' }, 'micSide' }, ...
    { { handles.sampleID; handles.sampleID; handles.sampleID; handles.sampleID; handles.sampleID; handles.sampleID }, 'sampleID' }, ...
    { { handles.driverChannel; handles.driverPreAmbientChannel; handles.driverPostAmbientChannel; handles.passengerChannel; handles.passengerPreAmbientChannel; handles.passengerPostAmbientChannel ], 'channel' }, ...
    { { 'signal'; 'preAmbient'; 'postAmbient'; 'signal'; 'preAmbient'; 'postAmbient' }, 'signalType' }, ...
    { { handles.driverSignalPathName; handles.driverPreAmbientPathName; handles.driverPostAmbientPathName; handles.passengerSignalPathName; handles.passengerPreAmbientPathName; handles.passengerPostAmbientPathName }, 'pathName' }, ...
    { { handles.driverSignalFileName; handles.driverPreAmbientFileName; handles.driverPostAmbientFileName; handles.passengerSignalFileName; handles.passengerPreAmbientFileName; handles.passengerPostAmbientFileName }, 'fileName' }, ...
    { { []; []; []; []; []; [] }, 'TriggerThreshold' }, ...
    { { []; []; []; []; []; [] }, 'idx' } );

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

if isempty( handles.data )
    handles.data = dataImport;
else
    handles.data = [ handles.data; dataImport ];
end

function handles = frontImport( handles )
% import front data into handles.data

opCond = handles.opCond;

frontImport = dataset( { { opCond; opCond; opCond }, 'opCond' }, ...
    { { 'front'; 'front'; 'front' }, 'micSide' }, ...
    { { handles.sampleID; handles.sampleID; handles.sampleID }, 'sampleID' }, ...
    { [ handles.frontChannel; handles.frontPreAmbientChannel; handles.frontPostAmbientChannel ], 'channel' }, ...
    { { 'signal'; 'preAmbient'; 'postAmbient' }, 'signalType' }, ...
    { { handles.frontSignalPathName; handles.frontPreAmbientPathName; handles.frontPostAmbientPathName }, 'pathName' }, ...
    { { handles.frontSignalFileName; handles.frontPreAmbientFileName; handles.frontPostAmbientFileName }, 'fileName' }, ...
```

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

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

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

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

opCond = handles.opCond;

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

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

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

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

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

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

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

```
        if strcmp( cellstr( opCond ), 'Stationary' )
            dat = handles.datStationary ;
            table = 'tableStationary' ;
        end

        if strcmp( cellstr( opCond ), '10 km/h' )
            dat = handles.dat10kmh ;
            table = 'table10kmh' ;
        end

        if strcmp( cellstr( opCond ), '20 km/h' )
            dat = handles.dat20kmh ;
            table = 'table20kmh' ;
        end

        if strcmp( cellstr( opCond ), '30 km/h' )
            dat = handles.dat30kmh ;
            table = 'table30kmh' ;
        end

        if strcmp( cellstr( opCond ), 'Reverse' )
            dat = handles.datReverse ;
            table = 'tableReverse' ;
        end

        set( handles.( table ), 'Data', dat );
    end
    % Enable the submit data button only if 4 samples exist for each operating condition
        if size (handles.datStationary, 1 ) >= 4 && size (handles.dat10kmh , 1 ) >= 4 && size (handles.dat20kmh , 1 ) >= 4 ...
                && size (handles.dat30kmh , 1 ) >= 4 && size (handles.datReverse , 1 ) >= 4
            set( handles.buttonSubmitData, 'Enable', 'on' );
            set( handles.textDataSubmittalMessage, 'String', ' ' );

        else
            set( handles.buttonSubmitData, 'Enable', 'off' );
            set( handles.textDataSubmittalMessage, 'String', 'Import at least four samples for each operating condition to submit data' );
        end
else
    sampleList = unique( handles.data( handles.data.opCond == opCond , { 'sampleID' } ) );
    sampleList = sort( sampleList.sampleID );

    if ~isempty( sampleList )

        for chalk = 1: length( sampleList )

            if isempty( handles.data( handles.data.opCond == opCond & ...
                    handles.data.sampleID == sampleList( chalk ) & ...
```

```
                handles.data.signalType == 'trigger', { 'fileName' } ) );
        else
            trigger = 'true';
            triggerSignal = handles.data( handles.data.opCond == opCond & ...
                handles.data.sampleID == sampleList( chalk ) & ...
                handles.data.signalType == 'trigger', { 'fileName' } ) ;
        end

        driverSignal = handles.data( handles.data.opCond == opCond & ...
            handles.data.sampleID == sampleList( chalk ) &  ...
            handles.data.micSide == 'driver' & ...
            handles.data.signalType == 'signal', { 'fileName' } );
        driverPreAmbient = handles.data( handles.data.opCond == opCond & ...
            handles.data.sampleID == sampleList( chalk ) &  ...
            handles.data.micSide == 'driver' & ...
            handles.data.signalType == 'preAmbient', { 'fileName' } );
        driverPostAmbient = handles.data( handles.data.opCond == opCond & ...
            handles.data.sampleID == sampleList( chalk ) &  ...
            handles.data.micSide == 'driver' & ...
            handles.data.signalType == 'postAmbient', { 'fileName' } );
        passengerSignal = handles.data( handles.data.opCond == opCond & ...
            handles.data.sampleID == sampleList( chalk ) &  ...
            handles.data.micSide == 'passenger' & ...
            handles.data.signalType == 'signal', { 'fileName' } );
        passengerPreAmbient = handles.data( handles.data.opCond == opCond & ...
            handles.data.sampleID == sampleList( chalk ) &  ...
            handles.data.micSide == 'passenger' & ...
            handles.data.signalType == 'preAmbient', { 'fileName' } );
        passengerPostAmbient = handles.data( handles.data.opCond == opCond & ...
            handles.data.sampleID == sampleList( chalk ) &  ...
            handles.data.micSide == 'passenger' & ...
            handles.data.signalType == 'postAmbient', { 'fileName' } );
        frontSignal = handles.data( handles.data.opCond == opCond & ...
            handles.data.sampleID == sampleList( chalk ) &  ...
            handles.data.micSide == 'front' & ...
            handles.data.signalType == 'signal', { 'fileName' } );
        frontPreAmbient = handles.data( handles.data.opCond == opCond & ...
            handles.data.sampleID == sampleList( chalk ) &  ...
            handles.data.micSide == 'front' & ...
            handles.data.signalType == 'preAmbient', { 'fileName' } );
        frontPostAmbient = handles.data( handles.data.opCond == opCond & ...
            handles.data.sampleID == sampleList( chalk ) &  ...
            handles.data.micSide == 'front' & ...
            handles.data.signalType == 'postAmbient', { 'fileName' } );

        if strcmp( cellstr( opCond ), 'Stationary' )
            dat( chalk, : ) = [ cellstr( sampleList( chalk ) ),...
                cellstr( driverSignal.fileName ), cellstr( driverPreAmbient.fileName ), cellstr(
                driverPostAmbient.fileName ),...
                cellstr( passengerSignal.fileName ), cellstr( passengerPreAmbient.fileName ), cellstr(
```

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

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


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

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

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

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

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

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

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

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

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

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

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

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

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

% --- Executes during object creation, after setting all properties.
```

```
function controlFrontSignalChannel_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

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

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

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

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

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

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

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

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

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

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

```
% 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

%***********************************************************************
%***********************************************************************
%      Empty Callbacks
%***********************************************************************
%***********************************************************************
function displayPassengerPreAmbientFile_Callback(hObject, eventdata, handles)
function displayDriverPreAmbientFile_Callback(hObject, eventdata, handles)
function displayDriverPostAmbientFile_Callback(hObject, eventdata, handles)
function displayFrontPreAmbientFile_Callback(hObject, eventdata, handles)
function displayPassengerPostAmbientFile_Callback(hObject, eventdata, handles)
function displayFrontPostAmbientFile_Callback(hObject, eventdata, handles)
```