```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' AggregateVehiclesClass - type info here...
'''
Public Class AggregateVehiclesClass

    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' MyData stores all the data required for a safety or a
    ' mobile6 class.
    '''
    Public Class MyData

        Public nVehicles As Integer          ' number of vehicles in this category
        Public Sales As Double               ' total sales for this category
        Public FuelEconomy As Double         ' fuel economy for this category
        Public FuelTankCapacity As Double    ' fuel tank capacity of this vehicle
        Public CurbWeight As Double          ' curb weight for this category
        Public InitialCurbWeight As Double   ' initial curb weight for this category
        Public DeltaCurbWeight As Double     ' curb weight - initial curb weight for this
    category
        Public Fleet() As Double             ' sales * survival
        Public CPM() As Double               ' the cpm for this category
        Public VMT() As Double               ' the vmt for this caregory
        Public Fatals() As Double            ' the fatalities for this category
        Public Gallons() As Double           ' the gallons for this category
        Public FuelTax() As Double           ' fuel tax revenue
        Public DiscFuelTax() As Double       ' discounted fuel tax revenue
        Public PreTaxFuel() As Double        ' pre-tax fuel expenditures
        Public DiscPreTaxFuel() As Double    ' discounted pre-tax fuel expenditures
        Public DriveSurplus() As Double      ' drive surplus
        Public DiscDriveSurplus() As Double  ' discounted drive surplus
        Public RefuelSurplus() As Double     ' refuel surplus
        Public DiscRefuelSurplus() As Double ' discounted refuel surplus
        Public Category As String            ' the name of this category
        Public CategoryIdx As Integer        ' the index of this category (in the array)

        Public Sub New()
            Me.nVehicles = 0
            Me.Sales = 0
            Me.FuelEconomy = 0
            Me.FuelTankCapacity = 0
            Me.CurbWeight = 0
            Me.InitialCurbWeight = 0
            Me.DeltaCurbWeight = 0
            ReDim Me.Fleet(25)
            ReDim Me.CPM(25)
            ReDim Me.VMT(25)
            ReDim Me.Fatals(25)
            ReDim Me.Gallons(25)
            ReDim Me.FuelTax(25)
            ReDim Me.DiscFuelTax(25)
            ReDim Me.PreTaxFuel(25)
            ReDim Me.DiscPreTaxFuel(25)
            ReDim Me.DriveSurplus(25)
            ReDim Me.DiscDriveSurplus(25)
            ReDim Me.RefuelSurplus(25)
            ReDim Me.DiscRefuelSurplus(25)
        End Sub

        Public Sub New(ByVal Category As String)
            Me.New()
            Me.Category = Category
        End Sub

    End Class
```

```
100     Public Enum Mobile6Category As Integer
101         UNKNOWN = 0             ' unknown category for vehicles without a mobile6 category
102         LDDV = 1
103         LDGV = 2
104         LDDT1 = 3
105         LDGT1 = 4
106         LDDT2 = 5
107         LDGT2 = 6
108         HDDV2b = 7
109         HDGV2b = 8
110         '
111         GAS_TOTAL = 9
112         DIESEL_TOTAL = 10
113         '
114         TOTAL = 11
115     End Enum
```

```
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169     Private Const nMobile6Classes = 11
170     Private Mobile6Classes() As MyData
171
172     Private Const nSafetyWeightClasses = 37
173     Private SafetyWeightClasses() As MyData
174
175     Private Totals As MyData
176
177     Private DBUG As Boolean = False
178
179     '''
180     ' the default constructor
181     '
182     Public Sub New()
183         Me.init()
184     End Sub
185
186     '''
187     ' the constructor with MY parameter
188     '
189     Public Sub New(ByVal Scenario As Integer, ByVal MY As Integer)
190         Dim MYIdx As Integer = CAFE.GMYI(MY)
191
192         If MYIdx < 0 Then
193             Me.exception("The entered model year '" & MY & "' is invalid or incorrect.")
194         Else
195             Me.init(Scenario, MYIdx)
196         End If
197     End Sub
198
199     '''
200     ' the initializer
201     '
202     Public Sub init()
203         ' initialize data to nothing
204         Me.SafetyClasses = Nothing
205         Me.Mobile6Classes = Nothing
```

```
206         Me.SafetyWeightClasses = Nothing
207         Me.Totals = Nothing
208     End Sub
209
210     '''
211     ' the initializer with a MYIdx parameter
212     '
213     Public Sub init(ByVal Scenario As Integer, ByVal MYIdx As Integer)
214         Me.init()
215         Me.AggregateSafetyClasses(Scenario, MYIdx)
216         Me.AggregateMobile6Classes(Scenario, MYIdx)
217         Me.AggregateSafetyWeightClasses(Scenario, MYIdx)
218         Me.AggregateTotals(Scenario, MYIdx)
219     End Sub
220
221
222  Methods for SafetyClasses
223
224
225  Methods for Mobile6Classes
226
227
228  Methods for SafetyWeightClasses
229
230
231  Methods for Totals
232
233     '''
234     ' method for debugging
235     '
236     Private Sub exception(ByVal msg As String)
237         If DBUG Then
238             Throw New System.Exception(msg)
239         End If
240
241         Debug.WriteLine("The following exception has occured: " & msg)
242     End Sub
243
244 End Class
245
```

```
1  Imports System.Reflection
2  Imports System.Runtime.InteropServices
3
4  ' General Information about an assembly is controlled through the following
5  ' set of attributes. Change these attribute values to modify the information
6  ' associated with an assembly.
7
8  ' Review the values of the assembly attributes
9
10 <Assembly: AssemblyTitle("")>
11 <Assembly: AssemblyDescription("")>
12 <Assembly: AssemblyCompany("U.S. Department of Transportation")>
13 <Assembly: AssemblyProduct("CAFE Compliance and Effects Model")>
14 <Assembly: AssemblyCopyright("None")>
15 <Assembly: AssemblyTrademark("")>
16 <Assembly: CLSCompliant(True)>
17
18 'The following GUID is for the ID of the typelib if this project is exposed to COM
19 <Assembly: Guid("DEAFFC4A-3D43-4129-B472-E86941E03CD7")>
20
21 ' Version information for an assembly consists of the following four values:
22 '
23 '      Major Version
24 '      Minor Version
25 '      Build Number
26 '      Revision
27 '
28 ' You can specify all the values or you can default the Build and Revision Numbers
29 ' by using the '*' as shown below:
30
31 <Assembly: AssemblyVersion("1.2.0.0")>
32
```

```
Public Class CAFE
    ' This class is used to contain 'global' instances of data and definitions.
    ' The definitions are 'shared', in other parlances this is called 'static' or 'global'
    '
    ' The Excel Application object...
    ' Public ExcelApp As Excel.Application
    Public Shared WeStartedExcel As Boolean
    Public Shared ModelRunning As Boolean
    Public Shared CAFE_Form As CafeForm

    Public Shared Profiling As CAFE_Reform.Profiling

    Public Shared ModelingCanceled As Boolean
    Public Shared ModelingComplete As Boolean
    Public Shared ModelingSuccessful As Boolean

    ' Debug aids. Keep track of the current scenario, current MY, current convergence loop, current MFG.
    ' When the value is out of scope (e.g. CurrentMfg after the MFG loop), set it to -1.
    Public Shared CurrentIteration As Integer
    Public Shared CurrentScenario As Integer
    Public Shared CurrentMY As Integer
    Public Shared CurrentConvergence As Integer
    Public Shared CurrentMfg As Integer

    ' Model start and end times
    Public Shared ModelStartTime As Date
    Public Shared ModelEndTime As Date
    Friend Shared SummaryOutput As SummaryInfoClass


    Public Enum PathsEnum
        Path1 = 1
        Path2 = 2
        Path3 = 3
    End Enum
    Public Enum LowAverageHigh
        Low = 1
        Average = 2
        High = 3
    End Enum
    Public Enum ApproachEnum
        LowCostFirst = 1
        EconomicEfficiency = 2
    End Enum
    Public Enum TechOverRide
        None = 0
        Skip = 1
        Force = 2
        Used = 3
    End Enum

    Public Structure OperatingModesStruct
        Public NoFines As Boolean
        Public Approach As ApproachEnum
        Public AutoToggleApproach As Boolean
        Public BackFillTechnology As Boolean
        Public StageAnalysis As Boolean
        Public MonteCarloAnalysis As Boolean
        Public MonteCarloIterations As Integer
        Public CreateEngineIfCarsInvolved As Boolean
        Public CreateEngineIfMultipleRC As Boolean
        Public CreateEngineIfTechEnableMissMatch As Boolean
        Public CreateTransmissionIfCarsInvolved As Boolean
        Public CreateTransmissionIfMultipleRC As Boolean
        Public CreateTransmissionIfTechEnableMissMatch As Boolean
        Public SkipIfAutoLightTruckMix As Boolean
        Public SkipIfMultiRC As Boolean
        Public SkipIfEnableMismatch As Boolean
        Public ReserveTechsIfInCompliance As Boolean
```

```
70      End Structure
71      Public Structure EncryptionSettingsStructure
72          ' file saving settings
73          Public EncryptOutputFiles As Boolean
74          Public EncryptionProvider As String
75          Public EncryptionKeyLength As Integer
76          Public EncryptDocumentProperties As Boolean
77          Public OutputPasswordToOpen As String
78          Public OutputPasswordToModify As String
79
80          ' file loading settings
81          Public ReadEncryptedInputFiles As Boolean
82          Public UseMultiplePasswords As Boolean
83          Public InputGlobalPassword As String
84          Public InputVehPassword As String
85          Public InputEngPassword As String
86          Public InputTranPassword As String
87          Public InputTechPassword As String
88      End Structure
89      Public Structure BaseLineInfoStructure
90          ' This structure holds information from the baseline (scenario 0)
91          Public FinalFuelEconomy() As Double         ' Fuel Economy for each vehicle
92      End Structure
93      ' stores the actual names of the encryption providers
94      Public Shared PasswordEncryptionProviders() As String = {"", _
95                           "Microsoft Enhanced Cryptographic Provider v1.0", _
96                           "Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider", ↙
    _
97                           "Microsoft Enhanced RSA and AES Cryptographic Provider (Prototype)" ↙
    , _
98                           "Microsoft RSA SChannel Cryptographic Provider", _
99                           "Microsoft Strong Cryptographic Provider" _
100                         }
101     Public Structure OutputSettingsStructure
102         Public DefaultOutputPath As String
103         Public HideUnusedRCs As Boolean
104
105         ' report settings
106         Public GenerateIndustryReport As Boolean
107         Public GenerateMfgsReport As Boolean
108         Public GenerateVehiclesReport As Boolean
109         Public GenerateEffectsReport As Boolean
110         Public GenerateEnvironmentalAssessmentReport As Boolean
111     End Structure
112     Public Structure TemplateSettingsStructure
113         Public UseTemplates As Boolean
114
115         Public DefaultTemplatePath As String
116         'Public LandscapeTemplatePath As String
117         'Public PortraitTemplatePath As String
118     End Structure
119     Public Structure OtherSettingsStrucutre
120         Public RunEffectsModel As Boolean
121         Public NoStartupWarning As Boolean
122         Public SingleFileDatasetInput As Boolean
123     End Structure
124
125     ' Model settings
126     Public Shared PathSettings As CAFE.PathsEnum
127     Public Shared FuelConsumptionSavingsEstimates As CAFE.LowAverageHigh
128     Public Shared TechnologyCostEstimates As CAFE.LowAverageHigh
129
130     Public Shared Technologies As VehicleTechnologiesClass
131     Public Shared Parameters As ParametersClass
132     Public Shared Scenarios As ScenariosClass
133     Public Shared Standards As New STD
134     Public Shared OperatingModes As New OperatingModesStruct
135     Public Shared EncryptionSettings As New EncryptionSettingsStructure
136     Public Shared OutputSettings As New OutputSettingsStructure
137     Public Shared TemplateSettings As New TemplateSettingsStructure
```

```
138     Public Shared OtherSettings As New OtherSettingsStrucutre
139
140     Public Shared MarketSimulation As New MarketSimulationClass
141
142     ' Regulatory Class Assignment
143     ' The first three RCs (0 - 2) are defined as for unregulated, domestic auto, and imported auto
144     ' the RCs from 3 and above are all light-trucks
145     Public Const RC_UnRegulated As Integer = 0
146     Public Const RC_DomesticAuto As Integer = 1
147     Public Const RC_ImportedAuto As Integer = 2
148
149     ' Putting Vehicles, Engines and Transmissions together...
150     ' "Master" are the intact copies right from the files. Non-master are the working copies
151     Public Shared MasterVehicles() As VehiclesClass
152     Public Shared MasterEngines() As EnginesClass
153     Public Shared MasterTransmissions() As TransmissionsClass
154     Public Shared MasterMfgs() As ManufacturerClass
155     Public Shared Vehicles() As VehiclesClass
156     Public Shared Engines() As EnginesClass
157     Public Shared Transmissions() As TransmissionsClass
158     Public Shared Mfgs() As ManufacturerClass
159     'Public Shared ModelYearsAvailable As ArrayList  ' Model & MSRP years in Vehicles file...
160     Public Shared ModelYearsSelected() As Boolean   ' If selected for modeling
161
162     ' Stacks to store data between tech applications
163     Public Shared VehiclesStack()() As VehiclesStackClass
164     Public Shared MfgsStack()() As ManufacturersStackClass
165     Public Shared IndustriesStack() As IndustriesWideStackClass
166     Public Shared EffectsStack() As EffectsStackClass
167     Public Shared EAsStack() As EAsStackClass           ' by scenario
168
169     ' store the aggregate data for safety and mobile6 classes
170     Public Shared ScenarioAggregateData() As AggregateVehiclesClass    ' stores current scenario data, variable
171     Public Shared BaselineAggregateData() As AggregateVehiclesClass    ' stores baseline scenario data, fixed
172     Public Shared SafetyEffectsLoaded As Boolean                       ' If safety effects worksheet was loaded
173     Public Shared SafetyClassesLoaded As Boolean                       ' If vehicles safety class column loaded
174     ' SafetyEffectsLoaded indicates if all the components needed for safety effects are on board.
175     ' if not, we won't report safety data.
176     Public Shared ReadOnly Property SafetyIncluded() As Boolean
177         Get
178             Return SafetyEffectsLoaded And SafetyClassesLoaded
179         End Get
180     End Property
181     ' specifies if NMNL worksheet has been loaded
182     Public Shared NMNLIncluded As Boolean
183
184     ' Constants that control termination of ConvergenceLoop
185     Public Const NumberOfAllowedConvergenceLoops As Integer = 10        ' Maximum number of loops
186     Public Const AllowedRcCAFEChange As Double = 0.01      ' change in any CAFE below which early termination allowed
187     Public Const AllowedRcShareChange As Double = 0.01     ' change in any RC share below which early termination allowed
188
189     'Baseline (scenario) info
190     Public Shared BaseLineInfo(-1) As BaseLineInfoStructure             ' For each MY
191
192 Model Year methods
193
194
195 Support Methods for Mfg() array
196
```

```
197
198  Remove NaNs methods
199
200
201  Functions for copying arrays, etc
202
203
204  Debug functions
205
206
207 End Class
208
```

```
1  Public Class CafeEffectsClass
2
3      ' define error values
4      Public Const COULD_NOT_OPEN_INPUT_FILE As Integer = -10
5      Public Const COULD_NOT_CALCULATE As Integer = -20
6      Public Const COULD_NOT_CREATE_RESULT_FILE As Integer = -30
7
8      Public Function crunchData(ByVal Scenario As Integer) As Long
9          ' CERTAIN PARAMETERS WILL BE TAKEN FROM THE TECH FILE OUTPUT:
10         ' FUEL ECONOMY STANDARDS, FUEL TYPE, BODY TYPE, CURB WEIGHT, GVWR, SALES(must be adjusted from param file forecasts)
11
12         ' step two is to crunch the data (now contained in vars) using the equations
13         'number of model years (total of model years, but arrays begin with zero)
14         Dim MY As Integer = CAFE.ModelYearsSelected.Length - 1
15         'number of calendar years (from year 0 to year 25)
16         'each array should begin where MY = CY
17         'all calculations must be done starting at the given MY and out 25 CYs (total of 26 CYs)
18         Dim CY As Integer = 25 + MY
19
20         'variables for 'for' loops
21         Dim i As Integer  'dimension i (rows) = Model Year
22         Dim j As Integer  'dimension j (columns) = Calendar Year
23         Dim k As Integer
24
25         ' Fuel economy standards, baseline and alternative
26         Dim actBaseStdLDGV(MY, CY) As Double
27         Dim actBaseStdLDDV(MY, CY) As Double
28         Dim actBaseStdLDGT1(MY, CY) As Double
29         Dim actBaseStdLDGT2(MY, CY) As Double
30         Dim actBaseStdLDDV1(MY, CY) As Double
31         Dim actBaseStdLDDV2(MY, CY) As Double
32         Dim actBaseStdHDGV2b(MY, CY) As Double
33         Dim actBaseStdHDDV2b(MY, CY) As Double
34
35         Dim actAltStdLDGV(MY, CY) As Double
36         Dim actAltStdLDDV(MY, CY) As Double
37         Dim actAltStdLDGT1(MY, CY) As Double
38         Dim actAltStdLDGT2(MY, CY) As Double
39         Dim actAltStdLDDV1(MY, CY) As Double
40         Dim actAltStdLDDV2(MY, CY) As Double
41         Dim actAltStdHDGV2b(MY, CY) As Double
42         Dim actAltStdHDDV2b(MY, CY) As Double
43
44         ' calculate base and alternative cost per mile (CPM) for each fuel
45         ' this does not need to be calculated on a per vehicle type basis
46         ' gasoline cost per mile for vehicles
47         ' base: under base CAFE standard
48         ' alternative: under alternative CAFE standard
49         ' base CPM vars:
50         Dim baseCPMLDGV(MY, CY) As Double
51         Dim baseCPMLDDV(MY, CY) As Double
52         Dim baseCPMLDGT1(MY, CY) As Double
53         Dim baseCPMLDGT2(MY, CY) As Double
54         Dim baseCPMLDDV1(MY, CY) As Double
55         Dim baseCPMLDDV2(MY, CY) As Double
56         Dim baseCPMHDGV2b(MY, CY) As Double
57         Dim baseCPMHDDV2b(MY, CY) As Double
58         'alternative CPM vars:
59         Dim altCPMLDGV(MY, CY) As Double
60         Dim altCPMLDDV(MY, CY) As Double
61         Dim altCPMLDGT1(MY, CY) As Double
62         Dim altCPMLDGT2(MY, CY) As Double
63         Dim altCPMLDDV1(MY, CY) As Double
64         Dim altCPMLDDV2(MY, CY) As Double
65         Dim altCPMHDGV2b(MY, CY) As Double
66         Dim altCPMHDDV2b(MY, CY) As Double
67
```

```
68          ' calculate base, rebound, alternative VMT for each
69          ' vehicle miles traveled by vehicles per calendar year
70          ' base: VMT under base CAFE standard
71          ' rebound: increase in VMT due to rebound effect
72          ' alternative: total annual VMT under alternative CAFE standard
73          ' base VMT vars:
74          Dim VMTLDGV(MY, CY) As Double
75          Dim VMTLDDV(MY, CY) As Double
76          Dim VMTLDGT1(MY, CY) As Double
77          Dim VMTLDGT2(MY, CY) As Double
78          Dim VMTLDDV1(MY, CY) As Double
79          Dim VMTLDDV2(MY, CY) As Double
80          Dim VMTHDGV2b(MY, CY) As Double
81          Dim VMTHDDV2b(MY, CY) As Double
82
83          'TOTALS
84          Dim gasVMTTotal(MY, CY) As Double
85          Dim diesVMTTotal(MY, CY) As Double
86
87          ' gallons vars:
88          Dim galLDGV(MY, CY) As Double
89          Dim galLDGT1(MY, CY) As Double
90          Dim galLDGT2(MY, CY) As Double
91          Dim galHDGV2b(MY, CY) As Double
92
93          Dim galLDDV(MY, CY) As Double
94          Dim galLDDV1(MY, CY) As Double
95          Dim galLDDV2(MY, CY) As Double
96          Dim galHDDV2b(MY, CY) As Double
97
98          'TOTALS:
99          Dim gasGalTotal(MY, CY) As Double
100         Dim diesGalTotal(MY, CY) As Double
101
102         ' load data for actAltStd, actBaseStd, altCPM, baseCPM, and VMT using the aggregate data
103         ' if we are running the baseline scenario (scenario 0), use CAFE.BaselineAggregateData
104         ' for alt and base values; otherwise, use CAFE.BaselineAggregateData for base values and
105         ' CAFE.ScenarioAggregateData for alt values
106         ''''''''''''''''''''''''''  LDDV  ''''''''''''''''''''''''''
107         Me.LoadMobile6AggregateArrays(Scenario, MY, CY, actBaseStdLDDV, actAltStdLDDV, baseCPMLDDV, altCPMLDDV, VMTLDDV, galLDDV, 1)
108
109         ''''''''''''''''''''''''''  LDGV  ''''''''''''''''''''''''''
110         Me.LoadMobile6AggregateArrays(Scenario, MY, CY, actBaseStdLDGV, actAltStdLDGV, baseCPMLDGV, altCPMLDGV, VMTLDGV, galLDGV, 2)
111
112         ''''''''''''''''''''''''''  LDDT1  ''''''''''''''''''''''''''
113         Me.LoadMobile6AggregateArrays(Scenario, MY, CY, actBaseStdLDDV1, actAltStdLDDV1, baseCPMLDDV1, altCPMLDDV1, VMTLDDV1, galLDDV1, 3)
114
115         ''''''''''''''''''''''''''  LDGT1  ''''''''''''''''''''''''''
116         Me.LoadMobile6AggregateArrays(Scenario, MY, CY, actBaseStdLDGT1, actAltStdLDGT1, baseCPMLDGT1, altCPMLDGT1, VMTLDGT1, galLDGT1, 4)
117
118         ''''''''''''''''''''''''''  LDDT2  ''''''''''''''''''''''''''
119         Me.LoadMobile6AggregateArrays(Scenario, MY, CY, actBaseStdLDDV2, actAltStdLDDV2, baseCPMLDDV2, altCPMLDDV2, VMTLDDV2, galLDDV2, 5)
120
121         ''''''''''''''''''''''''''  LDGT2  ''''''''''''''''''''''''''
122         Me.LoadMobile6AggregateArrays(Scenario, MY, CY, actBaseStdLDGT2, actAltStdLDGT2, baseCPMLDGT2, altCPMLDGT2, VMTLDGT2, galLDGT2, 6)
123
124         ''''''''''''''''''''''''''  HDDV2B  ''''''''''''''''''''''''''
125         Me.LoadMobile6AggregateArrays(Scenario, MY, CY, actBaseStdHDDV2b, actAltStdHDDV2b, baseCPMHDDV2b, altCPMHDDV2b, VMTHDDV2b, galHDDV2b, 7)
126
```

```
127         ''''''''''''''''''''''''''''  HDGV2B  ''''''''''''''''''''''''''''
128         Me.LoadMobile6AggregateArrays(Scenario, MY, CY, actBaseStdHDGV2b, actAltStdHDGV2b,
        baseCPMHDGV2b, altCPMHDGV2b, VMTHDGV2b, galHDGV2b, 8)
129
130         ' calculate external costs from additional vehicle use due to rebound effect
131         ' congestion vars
132         Dim congestionLDGV(MY, CY) As Double
133         Dim congestionLDGT1(MY, CY) As Double
134         Dim congestionLDGT2(MY, CY) As Double
135         Dim congestionHDGV2b(MY, CY) As Double
136
137         Dim congestionLDDV(MY, CY) As Double
138         Dim congestionLDDV1(MY, CY) As Double
139         Dim congestionLDDV2(MY, CY) As Double
140         Dim congestionHDDV2b(MY, CY) As Double
141
142         Dim discCongestionLDGV(MY, CY) As Double
143         Dim discCongestionLDGT1(MY, CY) As Double
144         Dim discCongestionLDGT2(MY, CY) As Double
145         Dim discCongestionHDGV2b(MY, CY) As Double
146
147         Dim discCongestionLDDV(MY, CY) As Double
148         Dim discCongestionLDDV1(MY, CY) As Double
149         Dim discCongestionLDDV2(MY, CY) As Double
150         Dim discCongestionHDDV2b(MY, CY) As Double
151
152         ' accidents vars
153         Dim accidentLDGV(MY, CY) As Double
154         Dim accidentLDGT1(MY, CY) As Double
155         Dim accidentLDGT2(MY, CY) As Double
156         Dim accidentHDGV2b(MY, CY) As Double
157
158         Dim accidentLDDV(MY, CY) As Double
159         Dim accidentLDDV1(MY, CY) As Double
160         Dim accidentLDDV2(MY, CY) As Double
161         Dim accidentHDDV2b(MY, CY) As Double
162
163         Dim discAccidentLDGV(MY, CY) As Double
164         Dim discAccidentLDGT1(MY, CY) As Double
165         Dim discAccidentLDGT2(MY, CY) As Double
166         Dim discAccidentHDGV2b(MY, CY) As Double
167
168         Dim discAccidentLDDV(MY, CY) As Double
169         Dim discAccidentLDDV1(MY, CY) As Double
170         Dim discAccidentLDDV2(MY, CY) As Double
171         Dim discAccidentHDDV2b(MY, CY) As Double
172
173         ' noise vars
174         Dim noiseLDGV(MY, CY) As Double
175         Dim noiseLDGT1(MY, CY) As Double
176         Dim noiseLDGT2(MY, CY) As Double
177         Dim noiseHDGV2b(MY, CY) As Double
178
179         Dim noiseLDDV(MY, CY) As Double
180         Dim noiseLDDV1(MY, CY) As Double
181         Dim noiseLDDV2(MY, CY) As Double
182         Dim noiseHDDV2b(MY, CY) As Double
183
184         Dim discNoiseLDGV(MY, CY) As Double
185         Dim discNoiseLDGT1(MY, CY) As Double
186         Dim discNoiseLDGT2(MY, CY) As Double
187         Dim discNoiseHDGV2b(MY, CY) As Double
188
189         Dim discNoiseLDDV(MY, CY) As Double
190         Dim discNoiseLDDV1(MY, CY) As Double
191         Dim discNoiseLDDV2(MY, CY) As Double
192         Dim discNoiseHDDV2b(MY, CY) As Double
193
194         ' total econ cost vars
```

```
195         Dim totalExternalLDGV(MY, CY) As Double
196         Dim totalExternalLDGT1(MY, CY) As Double
197         Dim totalExternalLDGT2(MY, CY) As Double
198         Dim totalExternalHDGV2b(MY, CY) As Double
199
200         Dim totalExternalLDDV(MY, CY) As Double
201         Dim totalExternalLDDV1(MY, CY) As Double
202         Dim totalExternalLDDV2(MY, CY) As Double
203         Dim totalExternalHDDV2b(MY, CY) As Double
204
205         'TOTALS
206         Dim gasCongestionTotal(MY, CY) As Double
207         Dim diesCongestionTotal(MY, CY) As Double
208         Dim gasAccidentTotal(MY, CY) As Double
209         Dim diesAccidentTotal(MY, CY) As Double
210         Dim gasNoiseTotal(MY, CY) As Double
211         Dim diesNoiseTotal(MY, CY) As Double
212         Dim gasTotalExternalTotal(MY, CY) As Double
213         Dim diesTotalExternalTotal(MY, CY) As Double
214
215         Dim discGasCongestionTotal(MY, CY) As Double
216         Dim discDiesCongestionTotal(MY, CY) As Double
217         Dim discGasAccidentTotal(MY, CY) As Double
218         Dim discDiesAccidentTotal(MY, CY) As Double
219         Dim discGasNoiseTotal(MY, CY) As Double
220         Dim discDiesNoiseTotal(MY, CY) As Double
221         Dim discGasTotalExternalTotal(MY, CY) As Double
222         Dim discDiesTotalExternalTotal(MY, CY) As Double
223
224         For i = 0 To MY
225             For j = 0 To CY
226                 If j >= i Then
227                     ' sum up vmt totals
228                     gasVMTTotal(i, j) = VMTLDGV(i, j) + VMTLDGT1(i, j) + VMTLDGT2(i, j) +
        VMTHDGV2b(i, j)
229                     diesVMTTotal(i, j) = VMTLDDV(i, j) + VMTLDDV1(i, j) + VMTLDDV2(i, j) +
        VMTHDDV2b(i, j)
230
231                     ' sum up gallons totals
232                     gasGalTotal(i, j) = galLDGV(i, j) + galLDGT1(i, j) + galLDGT2(i, j) +
        galHDGV2b(i, j)
233                     diesGalTotal(i, j) = galLDDV(i, j) + galLDDV1(i, j) + galLDDV2(i, j) +
        galHDDV2b(i, j)
234                 End If
235             Next j
236         Next i
237
238         For i = 0 To CY
239             For j = 0 To MY
240                 If i >= j Then
241                     ' calculate congestion cost
242                     congestionLDGV(j, i) = CAFE.Parameters.congestionCost * VMTLDGV(j, i)
243                     congestionLDGT1(j, i) = CAFE.Parameters.congestionCost * VMTLDGT1(j, i)
244                     congestionLDGT2(j, i) = CAFE.Parameters.congestionCost * VMTLDGT2(j, i)
245                     congestionHDGV2b(j, i) = CAFE.Parameters.congestionCost * VMTHDGV2b(j,
        i)
246
247                     congestionLDDV(j, i) = CAFE.Parameters.congestionCost * VMTLDDV(j, i)
248                     congestionLDDV1(j, i) = CAFE.Parameters.congestionCost * VMTLDDV1(j, i)
249                     congestionLDDV2(j, i) = CAFE.Parameters.congestionCost * VMTLDDV2(j, i)
250                     congestionHDDV2b(j, i) = CAFE.Parameters.congestionCost * VMTHDDV2b(j,
        i)
251
252                     'calculate discounted congestion cost
253                     discCongestionLDGV(j, i) = congestionLDGV(j, i) * ((1 + CAFE.Parameters.
        discountRate) ^ -i)
254                     discCongestionLDGT1(j, i) = congestionLDGT1(j, i) * ((1 + CAFE.
        Parameters.discountRate) ^ -i)
255                     discCongestionLDGT2(j, i) = congestionLDGT2(j, i) * ((1 + CAFE.
```

```
    Parameters.discountRate) ^ -i)
256                 discCongestionHDGV2b(j, i) = congestionHDGV2b(j, i) * ((1 + CAFE.
    Parameters.discountRate) ^ -i)
257
258                 discCongestionLDDV(j, i) = congestionLDDV(j, i) * ((1 + CAFE.Parameters.
    discountRate) ^ -i)
259                 discCongestionLDDV1(j, i) = congestionLDDV1(j, i) * ((1 + CAFE.
    Parameters.discountRate) ^ -i)
260                 discCongestionLDDV2(j, i) = congestionLDDV2(j, i) * ((1 + CAFE.
    Parameters.discountRate) ^ -i)
261                 discCongestionHDDV2b(j, i) = congestionHDDV2b(j, i) * ((1 + CAFE.
    Parameters.discountRate) ^ -i)
262
263                 'TOTALS:
264                 gasCongestionTotal(j, i) = congestionLDGV(j, i) + congestionLDGT1(j, i)
    + congestionLDGT2(j, i) + congestionHDGV2b(j, i)
265                 diesCongestionTotal(j, i) = congestionLDDV(j, i) + congestionLDDV1(j, i)
     + congestionLDDV2(j, i) + congestionHDDV2b(j, i)
266
267                 discGasCongestionTotal(j, i) = discCongestionLDGV(j, i) +
    discCongestionLDGT1(j, i) + discCongestionLDGT2(j, i) + discCongestionHDGV2b(j, i)
268                 discDiesCongestionTotal(j, i) = discCongestionLDDV(j, i) +
    discCongestionLDDV1(j, i) + discCongestionLDDV2(j, i) + discCongestionHDDV2b(j, i)
269
270                 ' calculate accident cost
271                 accidentLDGV(j, i) = CAFE.Parameters.accidentCost * VMTLDGV(j, i)
272                 accidentLDGT1(j, i) = CAFE.Parameters.accidentCost * VMTLDGT1(j, i)
273                 accidentLDGT2(j, i) = CAFE.Parameters.accidentCost * VMTLDGT2(j, i)
274                 accidentHDGV2b(j, i) = CAFE.Parameters.accidentCost * VMTHDGV2b(j, i)
275
276                 accidentLDDV(j, i) = CAFE.Parameters.accidentCost * VMTLDDV(j, i)
277                 accidentLDDV1(j, i) = CAFE.Parameters.accidentCost * VMTLDDV1(j, i)
278                 accidentLDDV2(j, i) = CAFE.Parameters.accidentCost * VMTLDDV2(j, i)
279                 accidentHDDV2b(j, i) = CAFE.Parameters.accidentCost * VMTHDDV2b(j, i)
280
281                 'calculate discounted accident cost
282                 discAccidentLDGV(j, i) = accidentLDGV(j, i) * ((1 + CAFE.Parameters.
    discountRate) ^ -i)
283                 discAccidentLDGT1(j, i) = accidentLDGT1(j, i) * ((1 + CAFE.Parameters.
    discountRate) ^ -i)
284                 discAccidentLDGT2(j, i) = accidentLDGT2(j, i) * ((1 + CAFE.Parameters.
    discountRate) ^ -i)
285                 discAccidentHDGV2b(j, i) = accidentHDGV2b(j, i) * ((1 + CAFE.Parameters.
    discountRate) ^ -i)
286
287                 discAccidentLDDV(j, i) = accidentLDDV(j, i) * ((1 + CAFE.Parameters.
    discountRate) ^ -i)
288                 discAccidentLDDV1(j, i) = accidentLDDV1(j, i) * ((1 + CAFE.Parameters.
    discountRate) ^ -i)
289                 discAccidentLDDV2(j, i) = accidentLDDV2(j, i) * ((1 + CAFE.Parameters.
    discountRate) ^ -i)
290                 discAccidentHDDV2b(j, i) = accidentHDDV2b(j, i) * ((1 + CAFE.Parameters.
    discountRate) ^ -i)
291
292                 'TOTALS:
293                 gasAccidentTotal(j, i) = accidentLDGV(j, i) + accidentLDGT1(j, i) +
    accidentLDGT2(j, i) + accidentHDGV2b(j, i)
294                 diesAccidentTotal(j, i) = accidentLDDV(j, i) + accidentLDDV1(j, i) +
    accidentLDDV2(j, i) + accidentHDDV2b(j, i)
295
296                 discGasAccidentTotal(j, i) = discAccidentLDGV(j, i) + discAccidentLDGT1
    (j, i) + discAccidentLDGT2(j, i) + discAccidentHDGV2b(j, i)
297                 discDiesAccidentTotal(j, i) = discAccidentLDDV(j, i) + discAccidentLDDV1
    (j, i) + discAccidentLDDV2(j, i) + discAccidentHDDV2b(j, i)
298
299                 ' calculate noise cost
300                 noiseLDGV(j, i) = CAFE.Parameters.noiseCost * VMTLDGV(j, i)
301                 noiseLDGT1(j, i) = CAFE.Parameters.noiseCost * VMTLDGT1(j, i)
302                 noiseLDGT2(j, i) = CAFE.Parameters.noiseCost * VMTLDGT2(j, i)
```

```
                    noiseHDGV2b(j, i) = CAFE.Parameters.noiseCost * VMTHDGV2b(j, i)

                    noiseLDDV(j, i) = CAFE.Parameters.noiseCost * VMTLDDV(j, i)
                    noiseLDDV1(j, i) = CAFE.Parameters.noiseCost * VMTLDDV1(j, i)
                    noiseLDDV2(j, i) = CAFE.Parameters.noiseCost * VMTLDDV2(j, i)
                    noiseHDDV2b(j, i) = CAFE.Parameters.noiseCost * VMTHDDV2b(j, i)

                    discNoiseLDGV(j, i) = noiseLDGV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                    discNoiseLDGT1(j, i) = noiseLDGT1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                    discNoiseLDGT2(j, i) = noiseLDGT2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                    discNoiseHDGV2b(j, i) = noiseHDGV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                    discNoiseLDDV(j, i) = noiseLDDV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                    discNoiseLDDV1(j, i) = noiseLDDV1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                    discNoiseLDDV2(j, i) = noiseLDDV2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                    discNoiseHDDV2b(j, i) = noiseHDDV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                    'TOTALS:
                    gasNoiseTotal(j, i) = noiseLDGV(j, i) + noiseLDGT1(j, i) + noiseLDGT2(j, i) + noiseHDGV2b(j, i)
                    diesNoiseTotal(j, i) = noiseLDDV(j, i) + noiseLDDV1(j, i) + noiseLDDV2(j, i) + noiseHDDV2b(j, i)

                    discGasNoiseTotal(j, i) = discNoiseLDGV(j, i) + discNoiseLDGT1(j, i) + discNoiseLDGT2(j, i) + discNoiseHDGV2b(j, i)
                    discDiesNoiseTotal(j, i) = discNoiseLDDV(j, i) + discNoiseLDDV1(j, i) + discNoiseLDDV2(j, i) + discNoiseHDDV2b(j, i)

                    ' Increase in Total Highway Use Externalities
                    ' total econ cost vars
                    totalExternalLDGV(j, i) = congestionLDGV(j, i) + accidentLDGV(j, i) + noiseLDGV(j, i)
                    totalExternalLDGT1(j, i) = congestionLDGT1(j, i) + accidentLDGT1(j, i) + noiseLDGT1(j, i)
                    totalExternalLDGT2(j, i) = congestionLDGT2(j, i) + accidentLDGT2(j, i) + noiseLDGT2(j, i)
                    totalExternalHDGV2b(j, i) = congestionHDGV2b(j, i) + accidentHDGV2b(j, i) + noiseHDGV2b(j, i)

                    totalExternalLDDV(j, i) = congestionLDDV(j, i) + accidentLDDV(j, i) + noiseLDGV(j, i)
                    totalExternalLDDV1(j, i) = congestionLDDV1(j, i) + accidentLDDV1(j, i) + noiseLDDV1(j, i)
                    totalExternalLDDV2(j, i) = congestionLDDV2(j, i) + accidentLDDV2(j, i) + noiseLDDV2(j, i)
                    totalExternalHDDV2b(j, i) = congestionHDDV2b(j, i) + accidentHDDV2b(j, i) + noiseHDDV2b(j, i)

                    'TOTALS:
                    gasTotalExternalTotal(j, i) = totalExternalLDGV(j, i) + totalExternalLDGT1(j, i) + totalExternalLDGT2(j, i) + totalExternalHDGV2b(j, i)
                    diesTotalExternalTotal(j, i) = totalExternalLDDV(j, i) + totalExternalLDDV1(j, i) + totalExternalLDDV2(j, i) + totalExternalHDDV2b(j, i)
                End If
            Next
        Next


        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        ' calculate fuel imports
```

```
349         ' fuel imports from alternative CAFE level and dollar value
350
351
352         ' fuel imports vars:
353
354         'finished gas imports
355         Dim finishImpLDGV(MY, CY) As Double
356         Dim finishImpLDGT1(MY, CY) As Double
357         Dim finishImpLDGT2(MY, CY) As Double
358         Dim finishImpHDGV2b(MY, CY) As Double
359
360         Dim finishImpLDDV(MY, CY) As Double
361         Dim finishImpLDDV1(MY, CY) As Double
362         Dim finishImpLDDV2(MY, CY) As Double
363         Dim finishImpHDDV2b(MY, CY) As Double
364
365         'crude petro imports
366         Dim crudeImpLDGV(MY, CY) As Double
367         Dim crudeImpLDGT1(MY, CY) As Double
368         Dim crudeImpLDGT2(MY, CY) As Double
369         Dim crudeImpHDGV2b(MY, CY) As Double
370
371         Dim crudeImpLDDV(MY, CY) As Double
372         Dim crudeImpLDDV1(MY, CY) As Double
373         Dim crudeImpLDDV2(MY, CY) As Double
374         Dim crudeImpHDDV2b(MY, CY) As Double
375
376         'dometic petro production
377         Dim prodDomLDGV(MY, CY) As Double
378         Dim prodDomLDGT1(MY, CY) As Double
379         Dim prodDomLDGT2(MY, CY) As Double
380         Dim prodDomHDGV2b(MY, CY) As Double
381
382         Dim prodDomLDDV(MY, CY) As Double
383         Dim prodDomLDDV1(MY, CY) As Double
384         Dim prodDomLDDV2(MY, CY) As Double
385         Dim prodDomHDDV2b(MY, CY) As Double
386
387         'TOTALS:
388         Dim gasFinishImpTotal(MY, CY) As Double
389         Dim diesFinishImpTotal(MY, CY) As Double
390         Dim gasCrudeImpTotal(MY, CY) As Double
391         Dim diesCrudeImpTotal(MY, CY) As Double
392         Dim gasProdDomTotal(MY, CY) As Double
393         Dim diesProdDomTotal(MY, CY) As Double
394
395         For i = 0 To CY
396             For j = 0 To MY
397                 If i >= j Then
398                     'finished gas imports
399                     finishImpLDGV(j, i) = (galLDGV(j, i) / 42) * CAFE.Parameters.gasLowImp
400                     finishImpLDGT1(j, i) = (galLDGT1(j, i) / 42) * CAFE.Parameters.gasLowImp
401                     finishImpLDGT2(j, i) = (galLDGT2(j, i) / 42) * CAFE.Parameters.gasLowImp
402                     finishImpHDGV2b(j, i) = (galHDGV2b(j, i) / 42) * CAFE.Parameters.
    gasLowImp
403
404                     finishImpLDDV(j, i) = (galLDDV(j, i) / 42) * CAFE.Parameters.diesLowImp
405                     finishImpLDDV1(j, i) = (galLDDV1(j, i) / 42) * CAFE.Parameters.
    diesLowImp
406                     finishImpLDDV2(j, i) = (galLDDV2(j, i) / 42) * CAFE.Parameters.
    diesLowImp
407                     finishImpHDDV2b(j, i) = (galHDDV2b(j, i) / 42) * CAFE.Parameters.
    diesLowImp
408
409                     CAFE.NanToZero(finishImpLDGV(j, i))
410                     CAFE.NanToZero(finishImpLDGT1(j, i))
411                     CAFE.NanToZero(finishImpLDGT2(j, i))
412                     CAFE.NanToZero(finishImpHDGV2b(j, i))
413                     CAFE.NanToZero(finishImpLDDV(j, i))
```

```
414                     CAFE.NanToZero(finishImpLDDV1(j, i))
415                     CAFE.NanToZero(finishImpLDDV2(j, i))
416                     CAFE.NanToZero(finishImpHDDV2b(j, i))
417
418                     'TOTALS:
419                     gasFinishImpTotal(j, i) = finishImpLDGV(j, i) + finishImpLDGT1(j, i) + finishImpLDGT2(j, i) + finishImpHDGV2b(j, i)
420                     diesFinishImpTotal(j, i) = finishImpLDDV(j, i) + finishImpLDDV1(j, i) + finishImpLDDV2(j, i) + finishImpHDDV2b(j, i)
421
422                     'crude petro imports
423                     crudeImpLDGV(j, i) = (galLDGV(j, i) / 42) * CAFE.Parameters.gasLowDomRef * CAFE.Parameters.gasLowDomRefImpCrude
424                     crudeImpLDGT1(j, i) = (galLDGT1(j, i) / 42) * CAFE.Parameters.gasLowDomRef * CAFE.Parameters.gasLowDomRefImpCrude
425                     crudeImpLDGT2(j, i) = (galLDGT2(j, i) / 42) * CAFE.Parameters.gasLowDomRef * CAFE.Parameters.gasLowDomRefImpCrude
426                     crudeImpHDGV2b(j, i) = (galHDGV2b(j, i) / 42) * CAFE.Parameters.gasLowDomRef * CAFE.Parameters.gasLowDomRefImpCrude
427
428                     crudeImpLDDV(j, i) = (galLDDV(j, i) / 42) * CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefImpCrude
429                     crudeImpLDDV1(j, i) = (galLDDV1(j, i) / 42) * CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefImpCrude
430                     crudeImpLDDV2(j, i) = (galLDDV2(j, i) / 42) * CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefImpCrude
431                     crudeImpHDDV2b(j, i) = (galHDDV2b(j, i) / 42) * CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefImpCrude
432
433                     CAFE.NanToZero(crudeImpLDGV(j, i))
434                     CAFE.NanToZero(crudeImpLDGT1(j, i))
435                     CAFE.NanToZero(crudeImpLDGT2(j, i))
436                     CAFE.NanToZero(crudeImpHDGV2b(j, i))
437                     CAFE.NanToZero(crudeImpLDDV(j, i))
438                     CAFE.NanToZero(crudeImpLDDV1(j, i))
439                     CAFE.NanToZero(crudeImpLDDV2(j, i))
440                     CAFE.NanToZero(crudeImpHDDV2b(j, i))
441
442                     'TOTALS:
443                     gasCrudeImpTotal(j, i) = crudeImpLDGV(j, i) + crudeImpLDGT1(j, i) + crudeImpLDGT2(j, i) + crudeImpHDGV2b(j, i)
444                     diesCrudeImpTotal(j, i) = crudeImpLDDV(j, i) + crudeImpLDDV1(j, i) + crudeImpLDDV2(j, i) + crudeImpHDDV2b(j, i)
445
446                     'dometic petro production from alternative
447                     prodDomLDGV(j, i) = (galLDGV(j, i) / 42) * CAFE.Parameters.gasLowDomRef * CAFE.Parameters.gasLowDomRefDomCrude
448                     prodDomLDGT1(j, i) = (galLDGT1(j, i) / 42) * CAFE.Parameters.gasLowDomRef * CAFE.Parameters.gasLowDomRefDomCrude
449                     prodDomLDGT2(j, i) = (galLDGT2(j, i) / 42) * CAFE.Parameters.gasLowDomRef * CAFE.Parameters.gasLowDomRefDomCrude
450                     prodDomHDGV2b(j, i) = (galHDGV2b(j, i) / 42) * CAFE.Parameters.gasLowDomRef * CAFE.Parameters.gasLowDomRefDomCrude
451
452                     prodDomLDDV(j, i) = (galLDDV(j, i) / 42) * CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefDomCrude
453                     prodDomLDDV1(j, i) = (galLDDV1(j, i) / 42) * CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefDomCrude
454                     prodDomLDDV2(j, i) = (galLDDV2(j, i) / 42) * CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefDomCrude
455                     prodDomHDDV2b(j, i) = (galHDDV2b(j, i) / 42) * CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefDomCrude
456
457                     CAFE.NanToZero(prodDomLDGV(j, i))
458                     CAFE.NanToZero(prodDomLDGT1(j, i))
459                     CAFE.NanToZero(prodDomLDGT2(j, i))
460                     CAFE.NanToZero(prodDomHDGV2b(j, i))
461                     CAFE.NanToZero(prodDomLDDV(j, i))
462                     CAFE.NanToZero(prodDomLDDV1(j, i))
```

```
463                     CAFE.NanToZero(prodDomLDDV2(j, i))
464                     CAFE.NanToZero(prodDomHDDV2b(j, i))
465
466                     'TOTALS:
467                     gasProdDomTotal(j, i) = prodDomLDGV(j, i) + prodDomLDGT1(j, i) +
    prodDomLDGT2(j, i) + prodDomHDGV2b(j, i)
468                     diesProdDomTotal(j, i) = prodDomLDDV(j, i) + prodDomLDDV1(j, i) +
    prodDomLDDV2(j, i) + prodDomHDDV2b(j, i)
469
470                 End If
471             Next
472         Next
473
474         ' calculate economic costs from oil imports
475
476         ' total economic cost vars
477         Dim totalEconCostLDGV(MY, CY) As Double
478         Dim totalEconCostLDGT1(MY, CY) As Double
479         Dim totalEconCostLDGT2(MY, CY) As Double
480         Dim totalEconCostHDGV2b(MY, CY) As Double
481
482         Dim totalEconCostLDDV(MY, CY) As Double
483         Dim totalEconCostLDDV1(MY, CY) As Double
484         Dim totalEconCostLDDV2(MY, CY) As Double
485         Dim totalEconCostHDDV2b(MY, CY) As Double
486
487         Dim discTotalEconCostLDGV(MY, CY) As Double
488         Dim discTotalEconCostLDGT1(MY, CY) As Double
489         Dim discTotalEconCostLDGT2(MY, CY) As Double
490         Dim discTotalEconCostHDGV2b(MY, CY) As Double
491
492         Dim discTotalEconCostLDDV(MY, CY) As Double
493         Dim discTotalEconCostLDDV1(MY, CY) As Double
494         Dim discTotalEconCostLDDV2(MY, CY) As Double
495         Dim discTotalEconCostHDDV2b(MY, CY) As Double
496
497         'TOTALS:
498         Dim gasTotalEconCostTotal(MY, CY) As Double
499         Dim diesTotalEconCostTotal(MY, CY) As Double
500
501         Dim discGasTotalEconCostTotal(MY, CY) As Double
502         Dim discDiesTotalEconCostTotal(MY, CY) As Double
503
504         For i = 0 To CY
505             For j = 0 To MY
506                 If i >= j Then
507                     totalEconCostLDGV(j, i) = ((finishImpLDGV(j, i) + crudeImpLDGV(j, i)) *
    CAFE.Parameters.totalEconomicCostBBL) + (prodDomLDGV(j, i) * CAFE.Parameters.
    monopsonyCost)
508                     totalEconCostLDGT1(j, i) = ((finishImpLDGT1(j, i) + crudeImpLDGT1(j, i))
     * CAFE.Parameters.totalEconomicCostBBL) + (prodDomLDGT1(j, i) * CAFE.Parameters.
    monopsonyCost)
509                     totalEconCostLDGT2(j, i) = ((finishImpLDGT2(j, i) + crudeImpLDGT2(j, i))
     * CAFE.Parameters.totalEconomicCostBBL) + (prodDomLDGT2(j, i) * CAFE.Parameters.
    monopsonyCost)
510                     totalEconCostHDGV2b(j, i) = ((finishImpHDGV2b(j, i) + crudeImpHDGV2b(j,
    i)) * CAFE.Parameters.totalEconomicCostBBL) + (prodDomHDGV2b(j, i) * CAFE.Parameters.
    monopsonyCost)
511
512                     totalEconCostLDDV(j, i) = ((finishImpLDDV(j, i) + crudeImpLDDV(j, i)) *
    CAFE.Parameters.totalEconomicCostBBL) + (prodDomLDDV(j, i) * CAFE.Parameters.
    monopsonyCost)
513                     totalEconCostLDDV1(j, i) = ((finishImpLDDV1(j, i) + crudeImpLDDV1(j, i))
     * CAFE.Parameters.totalEconomicCostBBL) + (prodDomLDDV1(j, i) * CAFE.Parameters.
    monopsonyCost)
514                     totalEconCostLDDV2(j, i) = ((finishImpLDDV2(j, i) + crudeImpLDDV2(j, i))
     * CAFE.Parameters.totalEconomicCostBBL) + (prodDomLDDV2(j, i) * CAFE.Parameters.
    monopsonyCost)
515                     totalEconCostHDDV2b(j, i) = ((finishImpHDDV2b(j, i) + crudeImpHDDV2b(j,
```

```
i)) * CAFE.Parameters.totalEconomicCostBBL) + (prodDomHDDV2b(j, i) * CAFE.Parameters.monopsonyCost)

                discTotalEconCostLDGV(j, i) = totalEconCostLDGV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                discTotalEconCostLDGT1(j, i) = totalEconCostLDGT1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                discTotalEconCostLDGT2(j, i) = totalEconCostLDGT2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                discTotalEconCostHDGV2b(j, i) = totalEconCostHDGV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                discTotalEconCostLDDV(j, i) = totalEconCostLDDV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                discTotalEconCostLDDV1(j, i) = totalEconCostLDDV1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                discTotalEconCostLDDV2(j, i) = totalEconCostLDDV2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                discTotalEconCostHDDV2b(j, i) = totalEconCostHDDV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                'TOTALS:
                gasTotalEconCostTotal(j, i) = totalEconCostLDGV(j, i) + totalEconCostLDGT1(j, i) + totalEconCostLDGT2(j, i) + totalEconCostHDGV2b(j, i)
                diesTotalEconCostTotal(j, i) = totalEconCostLDDV(j, i) + totalEconCostLDDV1(j, i) + totalEconCostLDDV2(j, i) + totalEconCostHDDV2b(j, i)

                discGasTotalEconCostTotal(j, i) = discTotalEconCostLDGV(j, i) + discTotalEconCostLDGT1(j, i) + discTotalEconCostLDGT2(j, i) + discTotalEconCostHDGV2b(j, i)
                discDiesTotalEconCostTotal(j, i) = discTotalEconCostLDDV(j, i) + discTotalEconCostLDDV1(j, i) + discTotalEconCostLDDV2(j, i) + discTotalEconCostHDDV2b(j, i)

            End If
        Next
    Next


    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    'CALCULATE CO2 EMISSIONS AND MONETIZED EMISSIONS:

    'average energy content (BTU/gal) var:
    Dim avgGasEng As Double
    Dim avgDiesEng As Double

    'calculate average energy content (BTU/gal):
    avgGasEng = (CAFE.Parameters.convGasEngDens * CAFE.Parameters.convGasMix) + (CAFE.Parameters.fedGasEngDens * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.caliGasEngDens * CAFE.Parameters.caliGasMix)
    avgDiesEng = (CAFE.Parameters.convDiesEngDens * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.lowDiesEngDens * CAFE.Parameters.lowDiesMix)

    'average mass density (grams/gal) var:
    Dim avgGasMassDens As Double
    Dim avgDiesMassDens As Double

    'calculate average mass density (grams/gal):
    avgGasMassDens = (CAFE.Parameters.convGasMassDens * CAFE.Parameters.convGasMix) + (CAFE.Parameters.fedGasMassDens * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.caliGasMassDens * CAFE.Parameters.caliGasMix)
    avgDiesMassDens = (CAFE.Parameters.convDiesMassDens * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.lowDiesMassDens * CAFE.Parameters.lowDiesMix)

    'average carbon content (%) var:
    Dim avgGasCarb As Double
    Dim avgDiesCarb As Double

```

```
562         'calculate average carbon content (%):
563         avgGasCarb = (CAFE.Parameters.convGasCarb * CAFE.Parameters.convGasMix) + (CAFE.
        Parameters.fedGasCarb * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.caliGasCarb * CAFE
        .Parameters.caliGasMix)
564         avgDiesCarb = (CAFE.Parameters.convDiesCarb * CAFE.Parameters.convDiesMix) + (CAFE.
        Parameters.lowDiesCarb * CAFE.Parameters.lowDiesMix)
565
566         'average upstream emissions (grams/MBTU) vars:
567         Dim avgUpEmisExtrCO2 As Double
568         Dim avgUpEmisTransCO2 As Double
569         Dim avgUpEmisRefCO2 As Double
570         Dim avgUpEmisTSDCO2 As Double
571
572         Dim avgUpEmisExtrDiesCO2 As Double
573         Dim avgUpEmisTransDiesCO2 As Double
574         Dim avgUpEmisRefDiesCO2 As Double
575         Dim avgUpEmisTSDDiesCO2 As Double
576
577         'calculate average upstream emissions (grams/MBTU)
578         avgUpEmisExtrCO2 = (CAFE.Parameters.totExtrConvGasCO2 * CAFE.Parameters.convGasMix)
        + (CAFE.Parameters.totExtrFedGasCO2 * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.
        totExtrCaliGasCO2 * CAFE.Parameters.caliGasMix)
579         avgUpEmisTransCO2 = (CAFE.Parameters.totTransConvGasCO2 * CAFE.Parameters.
        convGasMix) + (CAFE.Parameters.totTransFedGasCO2 * CAFE.Parameters.fedGasMix) + (CAFE.
        Parameters.totTransCaliGasCO2 * CAFE.Parameters.caliGasMix)
580         avgUpEmisRefCO2 = (CAFE.Parameters.totRefConvGasCO2 * CAFE.Parameters.convGasMix) +
        (CAFE.Parameters.totRefFedGasCO2 * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.
        totRefCaliGasCO2 * CAFE.Parameters.caliGasMix)
581         avgUpEmisTSDCO2 = (CAFE.Parameters.totTSDConvGasCO2 * CAFE.Parameters.convGasMix) +
        (CAFE.Parameters.totTSDFedGasCO2 * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.
        totTSDCaliGasCO2 * CAFE.Parameters.caliGasMix)
582
583         avgUpEmisExtrDiesCO2 = (CAFE.Parameters.totExtrConvDiesCO2 * CAFE.Parameters.
        convDiesMix) + (CAFE.Parameters.totExtrLowDiesCO2 * CAFE.Parameters.lowDiesMix)
584         avgUpEmisTransDiesCO2 = (CAFE.Parameters.totTransConvDiesCO2 * CAFE.Parameters.
        convDiesMix) + (CAFE.Parameters.totTransLowDiesCO2 * CAFE.Parameters.lowDiesMix)
585         avgUpEmisRefDiesCO2 = (CAFE.Parameters.totRefConvDiesCO2 * CAFE.Parameters.
        convDiesMix) + (CAFE.Parameters.totRefLowDiesCO2 * CAFE.Parameters.lowDiesMix)
586         avgUpEmisTSDDiesCO2 = (CAFE.Parameters.totTSDConvDiesCO2 * CAFE.Parameters.
        convDiesMix) + (CAFE.Parameters.totTSDLowDiesCO2 * CAFE.Parameters.lowDiesMix)
587
588         'adjusted average upstream emissions (grams/gal) vars:
589         Dim adjUpEmisExtrCO2 As Double
590         Dim adjUpEmisTransCO2 As Double
591         Dim adjUpEmisRefCO2 As Double
592         Dim adjUpEmisTSDCO2 As Double
593
594         Dim adjUpEmisExtrDiesCO2 As Double
595         Dim adjUpEmisTransDiesCO2 As Double
596         Dim adjUpEmisRefDiesCO2 As Double
597         Dim adjUpEmisTSDDiesCO2 As Double
598
599         'calculate adjusted average upstream emissions (grams/gal):
600         adjUpEmisExtrCO2 = avgUpEmisExtrCO2 * (avgGasEng / 10 ^ 6)
601         adjUpEmisTransCO2 = avgUpEmisTransCO2 * (avgGasEng / 10 ^ 6)
602         adjUpEmisRefCO2 = avgUpEmisRefCO2 * (avgGasEng / 10 ^ 6)
603         adjUpEmisTSDCO2 = avgUpEmisTSDCO2 * (avgGasEng / 10 ^ 6)
604
605         adjUpEmisExtrDiesCO2 = avgUpEmisExtrDiesCO2 * (avgDiesEng / 10 ^ 6)
606         adjUpEmisTransDiesCO2 = avgUpEmisTransDiesCO2 * (avgDiesEng / 10 ^ 6)
607         adjUpEmisRefDiesCO2 = avgUpEmisRefDiesCO2 * (avgDiesEng / 10 ^ 6)
608         adjUpEmisTSDDiesCO2 = avgUpEmisTSDDiesCO2 * (avgDiesEng / 10 ^ 6)
609
610         'Carbon Dioxide (CO2) Emissions from Domestic Crude Petroleum Extraction
611         Dim emisExtrCO2LDGV(MY, CY) As Double
612         Dim emisExtrCO2LDGT1(MY, CY) As Double
613         Dim emisExtrCO2LDGT2(MY, CY) As Double
614         Dim emisExtrCO2HDGV2b(MY, CY) As Double
615
```

```
616         Dim emisExtrCO2LDDV(MY, CY) As Double
617         Dim emisExtrCO2LDDV1(MY, CY) As Double
618         Dim emisExtrCO2LDDV2(MY, CY) As Double
619         Dim emisExtrCO2HDDV2b(MY, CY) As Double
620
621         For i = 0 To CY
622             For j = 0 To MY
623                 If i >= j Then
624                     emisExtrCO2LDGV(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
    gasLowDomRefDomCrude) * (adjUpEmisExtrCO2 * galLDGV(j, i)) / 10 ^ 6
625                     emisExtrCO2LDGT1(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
    .gasLowDomRefDomCrude) * (adjUpEmisExtrCO2 * galLDGT1(j, i)) / 10 ^ 6
626                     emisExtrCO2LDGT2(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
    .gasLowDomRefDomCrude) * (adjUpEmisExtrCO2 * galLDGT2(j, i)) / 10 ^ 6
627                     emisExtrCO2HDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.
    Parameters.gasLowDomRefDomCrude) * (adjUpEmisExtrCO2 * galHDGV2b(j, i)) / 10 ^ 6
628
629                     emisExtrCO2LDDV(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters
    .diesLowDomRefDomCrude) * (adjUpEmisExtrDiesCO2 * galLDDV(j, i)) / 10 ^ 6
630                     emisExtrCO2LDDV1(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
    Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesCO2 * galLDDV1(j, i)) / 10 ^ 6
631                     emisExtrCO2LDDV2(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
    Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesCO2 * galLDDV2(j, i)) / 10 ^ 6
632                     emisExtrCO2HDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
    Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesCO2 * galHDDV2b(j, i)) / 10 ^ 6
633                 End If
634             Next
635         Next
636
637         'Carbon Dioxide (CO2) Emissions from Domestic Crude Petroleum Transportation
638         Dim emisTransCO2LDGV(MY, CY) As Double
639         Dim emisTransCO2LDGT1(MY, CY) As Double
640         Dim emisTransCO2LDGT2(MY, CY) As Double
641         Dim emisTransCO2HDGV2b(MY, CY) As Double
642
643         Dim emisTransCO2LDDV(MY, CY) As Double
644         Dim emisTransCO2LDDV1(MY, CY) As Double
645         Dim emisTransCO2LDDV2(MY, CY) As Double
646         Dim emisTransCO2HDDV2b(MY, CY) As Double
647
648         For i = 0 To CY
649             For j = 0 To MY
650                 If i >= j Then
651                     emisTransCO2LDGV(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransCO2 * galLDGV(j, i)) / 10 ^ 6
652                     emisTransCO2LDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransCO2 * galLDGT1(j, i)) / 10 ^ 6
653                     emisTransCO2LDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransCO2 * galLDGT2(j, i)) / 10 ^ 6
654                     emisTransCO2HDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransCO2 * galHDGV2b(j, i)) / 10 ^ 6
655
656                     emisTransCO2LDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesCO2 * galLDDV(j, i)) / 10 ^ 6
657                     emisTransCO2LDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesCO2 * galLDDV1(j, i)) / 10 ^ 6
658                     emisTransCO2LDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesCO2 * galLDDV2(j, i)) / 10 ^ 6
659                     emisTransCO2HDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesCO2 * galHDDV2b(j, i)) / 10 ^ 6
660                 End If
661             Next
662         Next
663         ' Carbon Dioxide (CO2) Emissions from Domestic Crude Petroleum Refining to Produce
    Gasoline
664         Dim emisRefCO2LDGV(MY, CY) As Double
665         Dim emisRefCO2LDGT1(MY, CY) As Double
666         Dim emisRefCO2LDGT2(MY, CY) As Double
667         Dim emisRefCO2HDGV2b(MY, CY) As Double
```

```
668
669         Dim emisRefCO2LDDV(MY, CY) As Double
670         Dim emisRefCO2LDDV1(MY, CY) As Double
671         Dim emisRefCO2LDDV2(MY, CY) As Double
672         Dim emisRefCO2HDDV2b(MY, CY) As Double
673
674         For i = 0 To CY
675             For j = 0 To MY
676                 If i >= j Then
677                     emisRefCO2LDGV(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefCO2 ↙
     * galLDGV(j, i)) / 10 ^ 6
678                     emisRefCO2LDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) *                ↙
    (adjUpEmisRefCO2 * galLDGT1(j, i)) / 10 ^ 6
679                     emisRefCO2LDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) *                ↙
    (adjUpEmisRefCO2 * galLDGT2(j, i)) / 10 ^ 6
680                     emisRefCO2HDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) *               ↙
    (adjUpEmisRefCO2 * galHDGV2b(j, i)) / 10 ^ 6
681
682                     emisRefCO2LDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *                ↙
    (adjUpEmisRefDiesCO2 * galLDDV(j, i)) / 10 ^ 6
683                     emisRefCO2LDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *               ↙
    (adjUpEmisRefDiesCO2 * galLDDV1(j, i)) / 10 ^ 6
684                     emisRefCO2LDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *               ↙
    (adjUpEmisRefDiesCO2 * galLDDV2(j, i)) / 10 ^ 6
685                     emisRefCO2HDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *              ↙
    (adjUpEmisRefDiesCO2 * galHDDV2b(j, i)) / 10 ^ 6
686                 End If
687             Next
688         Next
689
690         'Carbon Dioxide (CO2) Emissions from Gasoline Transportation, Storage, and          ↙
    Distribution
691         Dim emisTSDCO2LDGV(MY, CY) As Double
692         Dim emisTSDCO2LDGT1(MY, CY) As Double
693         Dim emisTSDCO2LDGT2(MY, CY) As Double
694         Dim emisTSDCO2HDGV2b(MY, CY) As Double
695
696         Dim emisTSDCO2LDDV(MY, CY) As Double
697         Dim emisTSDCO2LDDV1(MY, CY) As Double
698         Dim emisTSDCO2LDDV2(MY, CY) As Double
699         Dim emisTSDCO2HDDV2b(MY, CY) As Double
700
701         For i = 0 To CY
702             For j = 0 To MY
703                 If i >= j Then
704                     emisTSDCO2LDGV(j, i) = (adjUpEmisTSDCO2 * galLDGV(j, i)) / 10 ^ 6
705                     emisTSDCO2LDGT1(j, i) = (adjUpEmisTSDCO2 * galLDGT1(j, i)) / 10 ^ 6
706                     emisTSDCO2LDGT2(j, i) = (adjUpEmisTSDCO2 * galLDGT2(j, i)) / 10 ^ 6
707                     emisTSDCO2HDGV2b(j, i) = (adjUpEmisTSDCO2 * galHDGV2b(j, i)) / 10 ^ 6
708
709                     emisTSDCO2LDDV(j, i) = (adjUpEmisTSDDiesCO2 * galLDDV(j, i)) / 10 ^ 6
710                     emisTSDCO2LDDV1(j, i) = (adjUpEmisTSDDiesCO2 * galLDDV1(j, i)) / 10 ^ 6
711                     emisTSDCO2LDDV2(j, i) = (adjUpEmisTSDDiesCO2 * galLDDV2(j, i)) / 10 ^ 6
712                     emisTSDCO2HDDV2b(j, i) = (adjUpEmisTSDDiesCO2 * galHDDV2b(j, i)) / 10 ^  ↙
    6
713                 End If
714             Next
715         Next
716
717         'Carbon Dioxide (CO2) Emissions from Vehicle Operation ("Tank to Wheel")
718         Dim emisOpCO2LDGV(MY, CY) As Double
719         Dim emisOpCO2LDGT1(MY, CY) As Double
720         Dim emisOpCO2LDGT2(MY, CY) As Double
721         Dim emisOpCO2HDGV2b(MY, CY) As Double
722
723         Dim emisOpCO2LDDV(MY, CY) As Double
724         Dim emisOpCO2LDDV1(MY, CY) As Double
725         Dim emisOpCO2LDDV2(MY, CY) As Double
726         Dim emisOpCO2HDDV2b(MY, CY) As Double
```

```
727
728         For i = 0 To CY
729             For j = 0 To MY
730                 If i >= j Then
731                     emisOpCO2LDGV(j, i) = galLDGV(j, i) * 10 ^ 6 * avgGasMassDens *
    avgGasCarb * 10 ^ -12 * 44 / 12
732                     emisOpCO2LDGT1(j, i) = galLDGT1(j, i) * 10 ^ 6 * avgGasMassDens *
    avgGasCarb * 10 ^ -12 * 44 / 12
733                     emisOpCO2LDGT2(j, i) = galLDGT2(j, i) * 10 ^ 6 * avgGasMassDens *
    avgGasCarb * 10 ^ -12 * 44 / 12
734                     emisOpCO2HDGV2b(j, i) = galHDGV2b(j, i) * 10 ^ 6 * avgGasMassDens *
    avgGasCarb * 10 ^ -12 * 44 / 12
735
736                     emisOpCO2LDDV(j, i) = galLDDV(j, i) * 10 ^ 6 * avgDiesMassDens *
    avgDiesCarb * 10 ^ -12 * 44 / 12
737                     emisOpCO2LDDV1(j, i) = galLDDV1(j, i) * 10 ^ 6 * avgDiesMassDens *
    avgDiesCarb * 10 ^ -12 * 44 / 12
738                     emisOpCO2LDDV2(j, i) = galLDDV2(j, i) * 10 ^ 6 * avgDiesMassDens *
    avgDiesCarb * 10 ^ -12 * 44 / 12
739                     emisOpCO2HDDV2b(j, i) = galHDDV2b(j, i) * 10 ^ 6 * avgDiesMassDens *
    avgDiesCarb * 10 ^ -12 * 44 / 12
740                 End If
741             Next
742         Next
743
744         'Total ("Well to Wheel") Carbon Dioxide (CO2) Emissions
745         Dim emisTotCO2LDGV(MY, CY) As Double
746         Dim emisTotCO2LDGT1(MY, CY) As Double
747         Dim emisTotCO2LDGT2(MY, CY) As Double
748         Dim emisTotCO2HDGV2b(MY, CY) As Double
749
750         Dim emisTotCO2LDDV(MY, CY) As Double
751         Dim emisTotCO2LDDV1(MY, CY) As Double
752         Dim emisTotCO2LDDV2(MY, CY) As Double
753         Dim emisTotCO2HDDV2b(MY, CY) As Double
754
755         'TOTALS:
756         Dim gasEmisTotCO2Total(MY, CY) As Double
757         Dim diesEmisTotCO2Total(MY, CY) As Double
758
759         For i = 0 To CY
760             For j = 0 To MY
761                 If i >= j Then
762                     emisTotCO2LDGV(j, i) = emisExtrCO2LDGV(j, i) + emisTransCO2LDGV(j, i) +
    emisRefCO2LDGV(j, i) + emisTSDCO2LDGV(j, i) + emisOpCO2LDGV(j, i)
763                     emisTotCO2LDGT1(j, i) = emisExtrCO2LDGT1(j, i) + emisTransCO2LDGT1(j, i)
     + emisRefCO2LDGT1(j, i) + emisTSDCO2LDGT1(j, i) + emisOpCO2LDGT1(j, i)
764                     emisTotCO2LDGT2(j, i) = emisExtrCO2LDGT2(j, i) + emisTransCO2LDGT2(j, i)
     + emisRefCO2LDGT2(j, i) + emisTSDCO2LDGT2(j, i) + emisOpCO2LDGT2(j, i)
765                     emisTotCO2HDGV2b(j, i) = emisExtrCO2HDGV2b(j, i) + emisTransCO2HDGV2b(j,
     i) + emisRefCO2HDGV2b(j, i) + emisTSDCO2HDGV2b(j, i) + emisOpCO2HDGV2b(j, i)
766
767                     emisTotCO2LDDV(j, i) = emisExtrCO2LDDV(j, i) + emisTransCO2LDDV(j, i) +
    emisRefCO2LDDV(j, i) + emisTSDCO2LDDV(j, i) + emisOpCO2LDDV(j, i)
768                     emisTotCO2LDDV1(j, i) = emisExtrCO2LDDV1(j, i) + emisTransCO2LDDV1(j, i)
     + emisRefCO2LDDV1(j, i) + emisTSDCO2LDDV1(j, i) + emisOpCO2LDDV1(j, i)
769                     emisTotCO2LDDV2(j, i) = emisExtrCO2LDDV2(j, i) + emisTransCO2LDDV2(j, i)
     + emisRefCO2LDDV2(j, i) + emisTSDCO2LDDV2(j, i) + emisOpCO2LDDV2(j, i)
770                     emisTotCO2HDDV2b(j, i) = emisExtrCO2HDDV2b(j, i) + emisTransCO2HDDV2b(j,
     i) + emisRefCO2HDDV2b(j, i) + emisTSDCO2HDDV2b(j, i) + emisOpCO2HDDV2b(j, i)
771
772                     'TOTALS:
773                     gasEmisTotCO2Total(j, i) = emisTotCO2LDGV(j, i) + emisTotCO2LDGT1(j, i)
    + emisTotCO2LDGT2(j, i) + emisTotCO2HDGV2b(j, i)
774                     diesEmisTotCO2Total(j, i) = emisTotCO2LDDV(j, i) + emisTotCO2LDDV1(j, i)
     + emisTotCO2LDDV2(j, i) + emisTotCO2HDDV2b(j, i)
775                 End If
776             Next
777         Next
```

```
778
779         'Economic Benefit/Cost of Carbon Dioxide (CO2) Emissions
780         Dim damageEmisTotCO2LDGV(MY, CY) As Double
781         Dim damageEmisTotCO2LDGT1(MY, CY) As Double
782         Dim damageEmisTotCO2LDGT2(MY, CY) As Double
783         Dim damageEmisTotCO2HDGV2b(MY, CY) As Double
784
785         Dim damageEmisTotCO2LDDV(MY, CY) As Double
786         Dim damageEmisTotCO2LDDV1(MY, CY) As Double
787         Dim damageEmisTotCO2LDDV2(MY, CY) As Double
788         Dim damageEmisTotCO2HDDV2b(MY, CY) As Double
789
790         Dim discDamageEmisTotCO2LDGV(MY, CY) As Double
791         Dim discDamageEmisTotCO2LDGT1(MY, CY) As Double
792         Dim discDamageEmisTotCO2LDGT2(MY, CY) As Double
793         Dim discDamageEmisTotCO2HDGV2b(MY, CY) As Double
794
795         Dim discDamageEmisTotCO2LDDV(MY, CY) As Double
796         Dim discDamageEmisTotCO2LDDV1(MY, CY) As Double
797         Dim discDamageEmisTotCO2LDDV2(MY, CY) As Double
798         Dim discDamageEmisTotCO2HDDV2b(MY, CY) As Double
799
800         'TOTALS:
801         Dim gasDamageEmisTotCO2Total(MY, CY) As Double
802         Dim diesDamageEmisTotCO2Total(MY, CY) As Double
803
804         Dim discGasDamageEmisTotCO2Total(MY, CY) As Double
805         Dim discDiesDamageEmisTotCO2Total(MY, CY) As Double
806
807         For i = 0 To CY
808             For j = 0 To MY
809                 If i >= j Then
810                     damageEmisTotCO2LDGV(j, i) = emisTotCO2LDGV(j, i) * CAFE.Parameters.
    damageCostCO2
811                     damageEmisTotCO2LDGT1(j, i) = emisTotCO2LDGT1(j, i) * CAFE.Parameters.
    damageCostCO2
812                     damageEmisTotCO2LDGT2(j, i) = emisTotCO2LDGT2(j, i) * CAFE.Parameters.
    damageCostCO2
813                     damageEmisTotCO2HDGV2b(j, i) = emisTotCO2HDGV2b(j, i) * CAFE.Parameters.
    damageCostCO2
814
815                     damageEmisTotCO2LDDV(j, i) = emisTotCO2LDDV(j, i) * CAFE.Parameters.
    damageCostCO2
816                     damageEmisTotCO2LDDV1(j, i) = emisTotCO2LDDV1(j, i) * CAFE.Parameters.
    damageCostCO2
817                     damageEmisTotCO2LDDV2(j, i) = emisTotCO2LDDV2(j, i) * CAFE.Parameters.
    damageCostCO2
818                     damageEmisTotCO2HDDV2b(j, i) = emisTotCO2HDDV2b(j, i) * CAFE.Parameters.
    damageCostCO2
819
820                     discDamageEmisTotCO2LDGV(j, i) = damageEmisTotCO2LDGV(j, i) * ((1 + CAFE
    .Parameters.discountRate) ^ -i)
821                     discDamageEmisTotCO2LDGT1(j, i) = damageEmisTotCO2LDGT1(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
822                     discDamageEmisTotCO2LDGT2(j, i) = damageEmisTotCO2LDGT2(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
823                     discDamageEmisTotCO2HDGV2b(j, i) = damageEmisTotCO2HDGV2b(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
824
825                     discDamageEmisTotCO2LDDV(j, i) = damageEmisTotCO2LDDV(j, i) * ((1 + CAFE
    .Parameters.discountRate) ^ -i)
826                     discDamageEmisTotCO2LDDV1(j, i) = damageEmisTotCO2LDDV1(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
827                     discDamageEmisTotCO2LDDV2(j, i) = damageEmisTotCO2LDDV2(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
828                     discDamageEmisTotCO2HDDV2b(j, i) = damageEmisTotCO2HDDV2b(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
829
830                     'TOTALS:
```

```
                    gasDamageEmisTotCO2Total(j, i) = damageEmisTotCO2LDGV(j, i) + damageEmisTotCO2LDGT1(j, i) + damageEmisTotCO2LDGT2(j, i) + damageEmisTotCO2HDGV2b(j, i)
                    diesDamageEmisTotCO2Total(j, i) = damageEmisTotCO2LDDV(j, i) + damageEmisTotCO2LDDV1(j, i) + damageEmisTotCO2LDDV2(j, i) + damageEmisTotCO2HDDV2b(j, i)

                    discGasDamageEmisTotCO2Total(j, i) = discDamageEmisTotCO2LDGV(j, i) + discDamageEmisTotCO2LDGT1(j, i) + discDamageEmisTotCO2LDGT2(j, i) + discDamageEmisTotCO2HDGV2b(j, i)
                    discDiesDamageEmisTotCO2Total(j, i) = discDamageEmisTotCO2LDDV(j, i) + discDamageEmisTotCO2LDDV1(j, i) + discDamageEmisTotCO2LDDV2(j, i) + discDamageEmisTotCO2HDDV2b(j, i)

                End If
            Next
        Next


        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

        'CALCULATE CRITERIA POLLUTANT EMISSIONS AND MONETIZED EMISSIONS ... WORKING ON THIS NOW
        'CARBON MONOXIDE (CO)
        'average CO upstream emissions (grams/MBTU) vars:
        Dim avgUpEmisExtrCO As Double
        Dim avgUpEmisTransCO As Double
        Dim avgUpEmisRefCO As Double
        Dim avgUpEmisTSDCO As Double

        Dim avgUpEmisExtrDiesCO As Double
        Dim avgUpEmisTransDiesCO As Double
        Dim avgUpEmisRefDiesCO As Double
        Dim avgUpEmisTSDDiesCO As Double

        'calculate CO average upstream emissions (grams/MBTU)
        avgUpEmisExtrCO = (CAFE.Parameters.totExtrConvGasCO * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totExtrFedGasCO * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totExtrCaliGasCO * CAFE.Parameters.caliGasMix)
        avgUpEmisTransCO = (CAFE.Parameters.totTransConvGasCO * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totTransFedGasCO * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totTransCaliGasCO * CAFE.Parameters.caliGasMix)
        avgUpEmisRefCO = (CAFE.Parameters.totRefConvGasCO * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totRefFedGasCO * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totRefCaliGasCO * CAFE.Parameters.caliGasMix)
        avgUpEmisTSDCO = (CAFE.Parameters.totTSDConvGasCO * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totTSDFedGasCO * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totTSDCaliGasCO * CAFE.Parameters.caliGasMix)

        avgUpEmisExtrDiesCO = (CAFE.Parameters.totExtrConvDiesCO * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.totExtrLowDiesCO * CAFE.Parameters.lowDiesMix)
        avgUpEmisTransDiesCO = (CAFE.Parameters.totTransConvDiesCO * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.totTransLowDiesCO * CAFE.Parameters.lowDiesMix)
        avgUpEmisRefDiesCO = (CAFE.Parameters.totRefConvDiesCO * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.totRefLowDiesCO * CAFE.Parameters.lowDiesMix)
        avgUpEmisTSDDiesCO = (CAFE.Parameters.totTSDConvDiesCO * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.totTSDLowDiesCO * CAFE.Parameters.lowDiesMix)

        'adjusted CO average upstream emissions (grams/gal) vars:
        Dim adjUpEmisExtrCO As Double
        Dim adjUpEmisTransCO As Double
        Dim adjUpEmisRefCO As Double
        Dim adjUpEmisTSDCO As Double

        Dim adjUpEmisExtrDiesCO As Double
        Dim adjUpEmisTransDiesCO As Double
        Dim adjUpEmisRefDiesCO As Double
        Dim adjUpEmisTSDDiesCO As Double

        'calculate CO adjusted average upstream emissions (grams/gal):
```

```
880         adjUpEmisExtrCO = avgUpEmisExtrCO * (avgGasEng / 10 ^ 6)
881         adjUpEmisTransCO = avgUpEmisTransCO * (avgGasEng / 10 ^ 6)
882         adjUpEmisRefCO = avgUpEmisRefCO * (avgGasEng / 10 ^ 6)
883         adjUpEmisTSDCO = avgUpEmisTSDCO * (avgGasEng / 10 ^ 6)
884
885         adjUpEmisExtrDiesCO = avgUpEmisExtrDiesCO * (avgDiesEng / 10 ^ 6)
886         adjUpEmisTransDiesCO = avgUpEmisTransDiesCO * (avgDiesEng / 10 ^ 6)
887         adjUpEmisRefDiesCO = avgUpEmisRefDiesCO * (avgDiesEng / 10 ^ 6)
888         adjUpEmisTSDDiesCO = avgUpEmisTSDDiesCO * (avgDiesEng / 10 ^ 6)
889
890         'Carbon Monoxide (CO) Emissions from Domestic Crude Petroleum Extraction
891         Dim emisExtrCOLDGV(MY, CY) As Double
892         Dim emisExtrCOLDGT1(MY, CY) As Double
893         Dim emisExtrCOLDGT2(MY, CY) As Double
894         Dim emisExtrCOHDGV2b(MY, CY) As Double
895
896         Dim emisExtrCOLDDV(MY, CY) As Double
897         Dim emisExtrCOLDDV1(MY, CY) As Double
898         Dim emisExtrCOLDDV2(MY, CY) As Double
899         Dim emisExtrCOHDDV2b(MY, CY) As Double
900
901         For i = 0 To CY
902             For j = 0 To MY
903                 If i >= j Then
904                     emisExtrCOLDGV(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
    gasLowDomRefDomCrude) * (adjUpEmisExtrCO * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
905                     emisExtrCOLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
    gasLowDomRefDomCrude) * (adjUpEmisExtrCO * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 *
    1000)
906                     emisExtrCOLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
    gasLowDomRefDomCrude) * (adjUpEmisExtrCO * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 *
    1000)
907                     emisExtrCOHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
    .gasLowDomRefDomCrude) * (adjUpEmisExtrCO * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 *
    1000)
908
909                     emisExtrCOLDDV(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters.
    diesLowDomRefDomCrude) * (adjUpEmisExtrDiesCO * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 *
    1000)
910                     emisExtrCOLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters
    .diesLowDomRefDomCrude) * (adjUpEmisExtrDiesCO * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000
    * 1000)
911                     emisExtrCOLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters
    .diesLowDomRefDomCrude) * (adjUpEmisExtrDiesCO * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000
    * 1000)
912                     emisExtrCOHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
    Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesCO * galHDDV2b(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
913                 End If
914             Next
915         Next
916
917         'Carbon Monoxide (CO) Emissions from Domestic Crude Petroleum Transportation
918         Dim emisTransCOLDGV(MY, CY) As Double
919         Dim emisTransCOLDGT1(MY, CY) As Double
920         Dim emisTransCOLDGT2(MY, CY) As Double
921         Dim emisTransCOHDGV2b(MY, CY) As Double
922
923         Dim emisTransCOLDDV(MY, CY) As Double
924         Dim emisTransCOLDDV1(MY, CY) As Double
925         Dim emisTransCOLDDV2(MY, CY) As Double
926         Dim emisTransCOHDDV2b(MY, CY) As Double
927
928         For i = 0 To CY
929             For j = 0 To MY
930                 If i >= j Then
931                     emisTransCOLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransCO * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
932                     emisTransCOLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) *
```

```
    (adjUpEmisTransCO * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
933                 emisTransCOLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransCO * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
934                 emisTransCOHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransCO * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
935
936                 emisTransCOLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesCO * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
937                 emisTransCOLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesCO * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
938                 emisTransCOLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesCO * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
939                 emisTransCOHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesCO * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
940             End If
941         Next
942     Next
943
944     ' Carbon Monoxide (CO) Emissions from Domestic Crude Petroleum Refining to Produce
    Gasoline
945     Dim emisRefCOLDGV(MY, CY) As Double
946     Dim emisRefCOLDGT1(MY, CY) As Double
947     Dim emisRefCOLDGT2(MY, CY) As Double
948     Dim emisRefCOHDGV2b(MY, CY) As Double
949
950     Dim emisRefCOLDDV(MY, CY) As Double
951     Dim emisRefCOLDDV1(MY, CY) As Double
952     Dim emisRefCOLDDV2(MY, CY) As Double
953     Dim emisRefCOHDDV2b(MY, CY) As Double
954
955     For i = 0 To CY
956         For j = 0 To MY
957             If i >= j Then
958                 emisRefCOLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefCO *
     galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
959                 emisRefCOLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefCO
    * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
960                 emisRefCOLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefCO
    * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
961                 emisRefCOHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefCO
     * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
962
963                 emisRefCOLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesCO * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
964                 emisRefCOLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesCO * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
965                 emisRefCOLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesCO * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
966                 emisRefCOHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesCO * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
967             End If
968         Next
969     Next
970
971     'Carbon Monoxide (CO) Emissions from Gasoline Transportation, Storage, and
    Distribution
972     Dim emisTSDCOLDGV(MY, CY) As Double
973     Dim emisTSDCOLDGT1(MY, CY) As Double
974     Dim emisTSDCOLDGT2(MY, CY) As Double
975     Dim emisTSDCOHDGV2b(MY, CY) As Double
976
977     Dim emisTSDCOLDDV(MY, CY) As Double
978     Dim emisTSDCOLDDV1(MY, CY) As Double
979     Dim emisTSDCOLDDV2(MY, CY) As Double
980     Dim emisTSDCOHDDV2b(MY, CY) As Double
981
982     For i = 0 To CY
983         For j = 0 To MY
984             If i >= j Then
```

```
985                     emisTSDCOLDGV(j, i) = (adjUpEmisTSDCO * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
986                     emisTSDCOLDGT1(j, i) = (adjUpEmisTSDCO * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
987                     emisTSDCOLDGT2(j, i) = (adjUpEmisTSDCO * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
988                     emisTSDCOHDGV2b(j, i) = (adjUpEmisTSDCO * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
989
990                     emisTSDCOLDDV(j, i) = (adjUpEmisTSDDiesCO * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
991                     emisTSDCOLDDV1(j, i) = (adjUpEmisTSDDiesCO * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
992                     emisTSDCOLDDV2(j, i) = (adjUpEmisTSDDiesCO * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
993                     emisTSDCOHDDV2b(j, i) = (adjUpEmisTSDDiesCO * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
994                 End If
995             Next
996         Next
997
998         'Carbon Monoxide (CO) Emissions from Vehicle Operation ("Tank to Wheel")
999
1000        ' calculate emission rates for gas = (Conv% * ConvRate) + ((1-Conv%) * RFGRate)
1001        Dim emisRateCOLDGV(MY, CY) As Double
1002        Dim emisRateCOLDGT1(MY, CY) As Double
1003        Dim emisRateCOLDGT2(MY, CY) As Double
1004        Dim emisRateCOHDGV2b(MY, CY) As Double
1005
1006        For i = 0 To CY
1007            For j = 0 To MY
1008                If i >= j Then
1009                    emisRateCOLDGV(j, i) = (CAFE.Parameters.coGasLDGV(j, Math.Min(i, 25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters.coGasRFGLDGV(j, Math.Min(i, 25)))
1010                    emisRateCOLDGT1(j, i) = (CAFE.Parameters.coGasLDGT1(j, Math.Min(i, 25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters.coGasRFGLDGT1(j, Math.Min(i, 25)))
1011                    emisRateCOLDGT2(j, i) = (CAFE.Parameters.coGasLDGT2(j, Math.Min(i, 25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters.coGasRFGLDGT2(j, Math.Min(i, 25)))
1012                    emisRateCOHDGV2b(j, i) = (CAFE.Parameters.coGasHDGV2b(j, Math.Min(i, 25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters.coGasRFGHDGV2b(j, Math.Min(i, 25)))
1013                End If
1014            Next j
1015        Next i
1016
1017        Dim emisOpCOLDGV(MY, CY) As Double
1018        Dim emisOpCOLDGT1(MY, CY) As Double
1019        Dim emisOpCOLDGT2(MY, CY) As Double
1020        Dim emisOpCOHDGV2b(MY, CY) As Double
1021
1022        Dim emisOpCOLDDV(MY, CY) As Double
1023        Dim emisOpCOLDDV1(MY, CY) As Double
1024        Dim emisOpCOLDDV2(MY, CY) As Double
1025        Dim emisOpCOHDDV2b(MY, CY) As Double
1026
1027        For i = 0 To CY
1028            For j = 0 To MY
1029                If i >= j Then
1030                    emisOpCOLDGV(j, i) = (VMTLDGV(j, i) * emisRateCOLDGV(j, i)) * (10 ^ 6) / (454 * 2000 * 1000)
1031                    emisOpCOLDGT1(j, i) = (VMTLDGT1(j, i) * emisRateCOLDGT1(j, i)) * (10 ^ 6) / (454 * 2000 * 1000)
1032                    emisOpCOLDGT2(j, i) = (VMTLDGT2(j, i) * emisRateCOLDGT2(j, i)) * (10 ^ 6) / (454 * 2000 * 1000)
1033                    emisOpCOHDGV2b(j, i) = (VMTHDGV2b(j, i) * emisRateCOHDGV2b(j, i)) * (10 ^ 6) / (454 * 2000 * 1000)
```

```
1034
1035                    emisOpCOLDDV(j, i) = (VMTLDDV(j, i) * CAFE.Parameters.coDiesLDDV(j, Math
    .Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1036                    emisOpCOLDDV1(j, i) = (VMTLDDV1(j, i) * CAFE.Parameters.coDiesLDDT1(j,
    Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1037                    emisOpCOLDDV2(j, i) = (VMTLDDV2(j, i) * CAFE.Parameters.coDiesLDDT2(j,
    Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1038                    emisOpCOHDDV2b(j, i) = (VMTHDDV2b(j, i) * CAFE.Parameters.coDiesHDDV2b(j
    , Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1039                End If
1040            Next
1041        Next
1042
1043        'Total ("Well to Wheel") Carbon Monoxide (CO) Emissions
1044        Dim emisTotCOLDGV(MY, CY) As Double
1045        Dim emisTotCOLDGT1(MY, CY) As Double
1046        Dim emisTotCOLDGT2(MY, CY) As Double
1047        Dim emisTotCOHDGV2b(MY, CY) As Double
1048
1049        Dim emisTotCOLDDV(MY, CY) As Double
1050        Dim emisTotCOLDDV1(MY, CY) As Double
1051        Dim emisTotCOLDDV2(MY, CY) As Double
1052        Dim emisTotCOHDDV2b(MY, CY) As Double
1053
1054        'TOTALS:
1055        Dim gasEmisTotCOTotal(MY, CY) As Double
1056        Dim diesEmisTotCOTotal(MY, CY) As Double
1057
1058        For i = 0 To CY
1059            For j = 0 To MY
1060                If i >= j Then
1061                    emisTotCOLDGV(j, i) = emisExtrCOLDGV(j, i) + emisTransCOLDGV(j, i) +
    emisRefCOLDGV(j, i) + emisTSDCOLDGV(j, i) + emisOpCOLDGV(j, i)
1062                    emisTotCOLDGT1(j, i) = emisExtrCOLDGT1(j, i) + emisTransCOLDGT1(j, i) +
    emisRefCOLDGT1(j, i) + emisTSDCOLDGT1(j, i) + emisOpCOLDGT1(j, i)
1063                    emisTotCOLDGT2(j, i) = emisExtrCOLDGT2(j, i) + emisTransCOLDGT2(j, i) +
    emisRefCOLDGT2(j, i) + emisTSDCOLDGT2(j, i) + emisOpCOLDGT2(j, i)
1064                    emisTotCOHDGV2b(j, i) = emisExtrCOHDGV2b(j, i) + emisTransCOHDGV2b(j, i)
     + emisRefCOHDGV2b(j, i) + emisTSDCOHDGV2b(j, i) + emisOpCOHDGV2b(j, i)
1065
1066                    emisTotCOLDDV(j, i) = emisExtrCOLDDV(j, i) + emisTransCOLDDV(j, i) +
    emisRefCOLDDV(j, i) + emisTSDCOLDDV(j, i) + emisOpCOLDDV(j, i)
1067                    emisTotCOLDDV1(j, i) = emisExtrCOLDDV1(j, i) + emisTransCOLDDV1(j, i) +
    emisRefCOLDDV1(j, i) + emisTSDCOLDDV1(j, i) + emisOpCOLDDV1(j, i)
1068                    emisTotCOLDDV2(j, i) = emisExtrCOLDDV2(j, i) + emisTransCOLDDV2(j, i) +
    emisRefCOLDDV2(j, i) + emisTSDCOLDDV2(j, i) + emisOpCOLDDV2(j, i)
1069                    emisTotCOHDDV2b(j, i) = emisExtrCOHDDV2b(j, i) + emisTransCOHDDV2b(j, i)
     + emisRefCOHDDV2b(j, i) + emisTSDCOHDDV2b(j, i) + emisOpCOHDDV2b(j, i)
1070
1071                    'TOTALS:
1072                    gasEmisTotCOTotal(j, i) = emisTotCOLDGV(j, i) + emisTotCOLDGT1(j, i) +
    emisTotCOLDGT2(j, i) + emisTotCOHDGV2b(j, i)
1073                    diesEmisTotCOTotal(j, i) = emisTotCOLDDV(j, i) + emisTotCOLDDV1(j, i) +
    emisTotCOLDDV2(j, i) + emisTotCOHDDV2b(j, i)
1074                End If
1075            Next
1076        Next
1077
1078        'Economic Benefit/Cost of Total Carbon Monoxide (CO) Emissions
1079        Dim damageEmisTotCOLDGV(MY, CY) As Double
1080        Dim damageEmisTotCOLDGT1(MY, CY) As Double
1081        Dim damageEmisTotCOLDGT2(MY, CY) As Double
1082        Dim damageEmisTotCOHDGV2b(MY, CY) As Double
1083
1084        Dim damageEmisTotCOLDDV(MY, CY) As Double
1085        Dim damageEmisTotCOLDDV1(MY, CY) As Double
1086        Dim damageEmisTotCOLDDV2(MY, CY) As Double
1087        Dim damageEmisTotCOHDDV2b(MY, CY) As Double
1088
```

```
1089        Dim discDamageEmisTotCOLDGV(MY, CY) As Double
1090        Dim discDamageEmisTotCOLDGT1(MY, CY) As Double
1091        Dim discDamageEmisTotCOLDGT2(MY, CY) As Double
1092        Dim discDamageEmisTotCOHDGV2b(MY, CY) As Double
1093
1094        Dim discDamageEmisTotCOLDDV(MY, CY) As Double
1095        Dim discDamageEmisTotCOLDDV1(MY, CY) As Double
1096        Dim discDamageEmisTotCOLDDV2(MY, CY) As Double
1097        Dim discDamageEmisTotCOHDDV2b(MY, CY) As Double
1098
1099        'TOTALS:
1100        Dim gasDamageEmisTotCOTotal(MY, CY) As Double
1101        Dim diesDamageEmisTotCOTotal(MY, CY) As Double
1102
1103        Dim discGasDamageEmisTotCOTotal(MY, CY) As Double
1104        Dim discDiesDamageEmisTotCOTotal(MY, CY) As Double .
1105
1106        For i = 0 To CY
1107            For j = 0 To MY
1108                If i >= j Then
1109                    damageEmisTotCOLDGV(j, i) = emisTotCOLDGV(j, i) * CAFE.Parameters.
    damageCostCO / 1000
1110                    damageEmisTotCOLDGT1(j, i) = emisTotCOLDGT1(j, i) * CAFE.Parameters.
    damageCostCO / 1000
1111                    damageEmisTotCOLDGT2(j, i) = emisTotCOLDGT2(j, i) * CAFE.Parameters.
    damageCostCO / 1000
1112                    damageEmisTotCOHDGV2b(j, i) = emisTotCOHDGV2b(j, i) * CAFE.Parameters.
    damageCostCO / 1000
1113
1114                    damageEmisTotCOLDDV(j, i) = emisTotCOLDDV(j, i) * CAFE.Parameters.
    damageCostCO / 1000
1115                    damageEmisTotCOLDDV1(j, i) = emisTotCOLDDV1(j, i) * CAFE.Parameters.
    damageCostCO / 1000
1116                    damageEmisTotCOLDDV2(j, i) = emisTotCOLDDV2(j, i) * CAFE.Parameters.
    damageCostCO / 1000
1117                    damageEmisTotCOHDDV2b(j, i) = emisTotCOHDDV2b(j, i) * CAFE.Parameters.
    damageCostCO / 1000
1118
1119                    discDamageEmisTotCOLDGV(j, i) = damageEmisTotCOLDGV(j, i) * ((1 + CAFE.
    Parameters.discountRate) ^ -i)
1120                    discDamageEmisTotCOLDGT1(j, i) = damageEmisTotCOLDGT1(j, i) * ((1 + CAFE
    .Parameters.discountRate) ^ -i)
1121                    discDamageEmisTotCOLDGT2(j, i) = damageEmisTotCOLDGT2(j, i) * ((1 + CAFE
    .Parameters.discountRate) ^ -i)
1122                    discDamageEmisTotCOHDGV2b(j, i) = damageEmisTotCOHDGV2b(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
1123
1124                    discDamageEmisTotCOLDDV(j, i) = damageEmisTotCOLDDV(j, i) * ((1 + CAFE.
    Parameters.discountRate) ^ -i)
1125                    discDamageEmisTotCOLDDV1(j, i) = damageEmisTotCOLDDV1(j, i) * ((1 + CAFE
    .Parameters.discountRate) ^ -i)
1126                    discDamageEmisTotCOLDDV2(j, i) = damageEmisTotCOLDDV2(j, i) * ((1 + CAFE
    .Parameters.discountRate) ^ -i)
1127                    discDamageEmisTotCOHDDV2b(j, i) = damageEmisTotCOHDDV2b(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
1128
1129                    'TOTALS:
1130                    gasDamageEmisTotCOTotal(j, i) = damageEmisTotCOLDGV(j, i) +
    damageEmisTotCOLDGT1(j, i) + damageEmisTotCOLDGT2(j, i) + damageEmisTotCOHDGV2b(j, i)
1131                    diesDamageEmisTotCOTotal(j, i) = damageEmisTotCOLDDV(j, i) +
    damageEmisTotCOLDDV1(j, i) + damageEmisTotCOLDDV2(j, i) + damageEmisTotCOHDDV2b(j, i)
1132
1133                    discGasDamageEmisTotCOTotal(j, i) = discDamageEmisTotCOLDGV(j, i) +
    discDamageEmisTotCOLDGT1(j, i) + discDamageEmisTotCOLDGT2(j, i) +
    discDamageEmisTotCOHDGV2b(j, i)
1134                    discDiesDamageEmisTotCOTotal(j, i) = discDamageEmisTotCOLDDV(j, i) +
    discDamageEmisTotCOLDDV1(j, i) + discDamageEmisTotCOLDDV2(j, i) +
    discDamageEmisTotCOHDDV2b(j, i)
1135
```

```
                End If
            Next
        Next


        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        'VOLATILE ORGANIC COMPOUNDS (VOC)
        'average VOC upstream emissions (grams/MBTU) vars:
        Dim avgUpEmisExtrVOC As Double
        Dim avgUpEmisTransVOC As Double
        Dim avgUpEmisRefVOC As Double
        Dim avgUpEmisTSDVOC As Double

        Dim avgUpEmisExtrDiesVOC As Double
        Dim avgUpEmisTransDiesVOC As Double
        Dim avgUpEmisRefDiesVOC As Double
        Dim avgUpEmisTSDDiesVOC As Double

        'calculate VOC average upstream emissions (grams/MBTU)
        avgUpEmisExtrVOC = (CAFE.Parameters.totExtrConvGasVOC * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totExtrFedGasVOC * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totExtrCaliGasVOC * CAFE.Parameters.caliGasMix)
        avgUpEmisTransVOC = (CAFE.Parameters.totTransConvGasVOC * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totTransFedGasVOC * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totTransCaliGasVOC * CAFE.Parameters.caliGasMix)
        avgUpEmisRefVOC = (CAFE.Parameters.totRefConvGasVOC * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totRefFedGasVOC * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totRefCaliGasVOC * CAFE.Parameters.caliGasMix)
        avgUpEmisTSDVOC = (CAFE.Parameters.totTSDConvGasVOC * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totTSDFedGasVOC * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totTSDCaliGasVOC * CAFE.Parameters.caliGasMix)

        avgUpEmisExtrDiesVOC = (CAFE.Parameters.totExtrConvDiesVOC * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.totExtrLowDiesVOC * CAFE.Parameters.lowDiesMix)
        avgUpEmisTransDiesVOC = (CAFE.Parameters.totTransConvDiesVOC * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.totTransLowDiesVOC * CAFE.Parameters.lowDiesMix)
        avgUpEmisRefDiesVOC = (CAFE.Parameters.totRefConvDiesVOC * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.totRefLowDiesVOC * CAFE.Parameters.lowDiesMix)
        avgUpEmisTSDDiesVOC = (CAFE.Parameters.totTSDConvDiesVOC * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.totTSDLowDiesVOC * CAFE.Parameters.lowDiesMix)

        'adjusted VOC average upstream emissions (grams/gal) vars:
        Dim adjUpEmisExtrVOC As Double
        Dim adjUpEmisTransVOC As Double
        Dim adjUpEmisRefVOC As Double
        Dim adjUpEmisTSDVOC As Double

        Dim adjUpEmisExtrDiesVOC As Double
        Dim adjUpEmisTransDiesVOC As Double
        Dim adjUpEmisRefDiesVOC As Double
        Dim adjUpEmisTSDDiesVOC As Double

        'calculate VOC adjusted average upstream emissions (grams/gal):
        adjUpEmisExtrVOC = avgUpEmisExtrVOC * (avgGasEng / 10 ^ 6)
        adjUpEmisTransVOC = avgUpEmisTransVOC * (avgGasEng / 10 ^ 6)
        adjUpEmisRefVOC = avgUpEmisRefVOC * (avgGasEng / 10 ^ 6)
        adjUpEmisTSDVOC = avgUpEmisTSDVOC * (avgGasEng / 10 ^ 6)

        adjUpEmisExtrDiesVOC = avgUpEmisExtrDiesVOC * (avgDiesEng / 10 ^ 6)
        adjUpEmisTransDiesVOC = avgUpEmisTransDiesVOC * (avgDiesEng / 10 ^ 6)
        adjUpEmisRefDiesVOC = avgUpEmisRefDiesVOC * (avgDiesEng / 10 ^ 6)
        adjUpEmisTSDDiesVOC = avgUpEmisTSDDiesVOC * (avgDiesEng / 10 ^ 6)

        'Volatile Organic Compounds (VOC) Emissions from Domestic Crude Petroleum Extraction
        Dim emisExtrVOCLDGV(MY, CY) As Double
        Dim emisExtrVOCLDGT1(MY, CY) As Double
        Dim emisExtrVOCLDGT2(MY, CY) As Double
```

```
1191        Dim emisExtrVOCHDGV2b(MY, CY) As Double
1192
1193        Dim emisExtrVOCLDDV(MY, CY) As Double
1194        Dim emisExtrVOCLDDV1(MY, CY) As Double
1195        Dim emisExtrVOCLDDV2(MY, CY) As Double
1196        Dim emisExtrVOCHDDV2b(MY, CY) As Double
1197
1198        For i = 0 To CY
1199            For j = 0 To MY
1200                If i >= j Then
1201                    emisExtrVOCLDGV(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
        gasLowDomRefDomCrude) * (adjUpEmisExtrVOC * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 *
        1000)
1202                    emisExtrVOCLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
        .gasLowDomRefDomCrude) * (adjUpEmisExtrVOC * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 *
        1000)
1203                    emisExtrVOCLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
        .gasLowDomRefDomCrude) * (adjUpEmisExtrVOC * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 *
        1000)
1204                    emisExtrVOCHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.
        Parameters.gasLowDomRefDomCrude) * (adjUpEmisExtrVOC * galHDGV2b(j, i) * 10 ^ 6) / (454
        * 2000 * 1000)
1205
1206                    emisExtrVOCLDDV(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters
        .diesLowDomRefDomCrude) * (adjUpEmisExtrDiesVOC * galLDDV(j, i) * 10 ^ 6) / (454 * 2000
        * 1000)
1207                    emisExtrVOCLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
        Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesVOC * galLDDV1(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
1208                    emisExtrVOCLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
        Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesVOC * galLDDV2(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
1209                    emisExtrVOCHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
        Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesVOC * galHDDV2b(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
1210                End If
1211            Next
1212        Next
1213
1214        'Volatile Organic Compounds (VOC) Emissions from Domestic Crude Petroleum
        Transportation
1215        Dim emisTransVOCLDGV(MY, CY) As Double
1216        Dim emisTransVOCLDGT1(MY, CY) As Double
1217        Dim emisTransVOCLDGT2(MY, CY) As Double
1218        Dim emisTransVOCHDGV2b(MY, CY) As Double
1219
1220        Dim emisTransVOCLDDV(MY, CY) As Double
1221        Dim emisTransVOCLDDV1(MY, CY) As Double
1222        Dim emisTransVOCLDDV2(MY, CY) As Double
1223        Dim emisTransVOCHDDV2b(MY, CY) As Double
1224
1225        For i = 0 To CY
1226            For j = 0 To MY
1227                If i >= j Then
1228                    emisTransVOCLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) *
        (adjUpEmisTransVOC * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1229                    emisTransVOCLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) *
        (adjUpEmisTransVOC * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1230                    emisTransVOCLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) *
        (adjUpEmisTransVOC * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1231                    emisTransVOCHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) *
        (adjUpEmisTransVOC * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1232
1233                    emisTransVOCLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
        (adjUpEmisTransDiesVOC * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1234                    emisTransVOCLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
        (adjUpEmisTransDiesVOC * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1235                    emisTransVOCLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
        (adjUpEmisTransDiesVOC * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
```

```
1236                    emisTransVOCHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesVOC * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1237                End If
1238            Next
1239        Next
1240
1241        ' Volatile Organic Compounds (VOC) Emissions from Domestic Crude Petroleum Refining
    to Produce Gasoline
1242        Dim emisRefVOCLDGV(MY, CY) As Double
1243        Dim emisRefVOCLDGT1(MY, CY) As Double
1244        Dim emisRefVOCLDGT2(MY, CY) As Double
1245        Dim emisRefVOCHDGV2b(MY, CY) As Double
1246
1247        Dim emisRefVOCLDDV(MY, CY) As Double
1248        Dim emisRefVOCLDDV1(MY, CY) As Double
1249        Dim emisRefVOCLDDV2(MY, CY) As Double
1250        Dim emisRefVOCHDDV2b(MY, CY) As Double
1251
1252        For i = 0 To CY
1253            For j = 0 To MY
1254                If i >= j Then
1255                    emisRefVOCLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefVOC
     * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1256                    emisRefVOCLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisRefVOC * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1257                    emisRefVOCLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisRefVOC * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1258                    emisRefVOCHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisRefVOC * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1259
1260                    emisRefVOCLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesVOC * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1261                    emisRefVOCLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesVOC * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1262                    emisRefVOCLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesVOC * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1263                    emisRefVOCHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesVOC * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1264                End If
1265            Next
1266        Next
1267
1268        'Volatile Organic Compounds (VOC) Emissions from Gasoline Transportation, Storage,
    and Distribution
1269        Dim emisTSDVOCLDGV(MY, CY) As Double
1270        Dim emisTSDVOCLDGT1(MY, CY) As Double
1271        Dim emisTSDVOCLDGT2(MY, CY) As Double
1272        Dim emisTSDVOCHDGV2b(MY, CY) As Double
1273
1274        Dim emisTSDVOCLDDV(MY, CY) As Double
1275        Dim emisTSDVOCLDDV1(MY, CY) As Double
1276        Dim emisTSDVOCLDDV2(MY, CY) As Double
1277        Dim emisTSDVOCHDDV2b(MY, CY) As Double
1278
1279        For i = 0 To CY
1280            For j = 0 To MY
1281                If i >= j Then
1282                    emisTSDVOCLDGV(j, i) = (adjUpEmisTSDVOC * galLDGV(j, i) * 10 ^ 6) / (454
     * 2000 * 1000)
1283                    emisTSDVOCLDGT1(j, i) = (adjUpEmisTSDVOC * galLDGT1(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1284                    emisTSDVOCLDGT2(j, i) = (adjUpEmisTSDVOC * galLDGT2(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1285                    emisTSDVOCHDGV2b(j, i) = (adjUpEmisTSDVOC * galHDGV2b(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1286
1287                    emisTSDVOCLDDV(j, i) = (adjUpEmisTSDDiesVOC * galLDDV(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1288                    emisTSDVOCLDDV1(j, i) = (adjUpEmisTSDDiesVOC * galLDDV1(j, i) * 10 ^ 6)
```

```
    / (454 * 2000 * 1000)
1289                    emisTSDVOCLDDV2(j, i) = (adjUpEmisTSDDiesVOC * galLDDV2(j, i) * 10 ^ 6)
    / (454 * 2000 * 1000)
1290                    emisTSDVOCHDDV2b(j, i) = (adjUpEmisTSDDiesVOC * galHDDV2b(j, i) * 10 ^
    6) / (454 * 2000 * 1000)
1291                End If
1292            Next
1293        Next
1294
1295
1296        ''Volatile Organic Compounds (VOC) Emissions from Vehicle Operation ("Tank to Wheel"
    )
1297        ' calculate emission rates for gas = (Conv% * ConvRate) + ((1-Conv%) * RFGRate)
1298        Dim emisRateVOCLDGV(MY, CY) As Double
1299        Dim emisRateVOCLDGT1(MY, CY) As Double
1300        Dim emisRateVOCLDGT2(MY, CY) As Double
1301        Dim emisRateVOCHDGV2b(MY, CY) As Double
1302
1303        For i = 0 To CY
1304            For j = 0 To MY
1305                If i >= j Then
1306                    emisRateVOCLDGV(j, i) = (CAFE.Parameters.vocGasLDGV(j, Math.Min(i, 25))
    * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters.
    vocGasRFGLDGV(j, Math.Min(i, 25)))
1307                    emisRateVOCLDGT1(j, i) = (CAFE.Parameters.vocGasLDGT1(j, Math.Min(i,
    25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters
    .vocGasRFGLDGT1(j, Math.Min(i, 25)))
1308                    emisRateVOCLDGT2(j, i) = (CAFE.Parameters.vocGasLDGT2(j, Math.Min(i,
    25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters
    .vocGasRFGLDGT2(j, Math.Min(i, 25)))
1309                    emisRateVOCHDGV2b(j, i) = (CAFE.Parameters.vocGasHDGV2b(j, Math.Min(i,
    25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters
    .vocGasRFGHDGV2b(j, Math.Min(i, 25)))
1310                End If
1311            Next j
1312        Next i
1313
1314        Dim emisOpVOCLDGV(MY, CY) As Double
1315        Dim emisOpVOCLDGT1(MY, CY) As Double
1316        Dim emisOpVOCLDGT2(MY, CY) As Double
1317        Dim emisOpVOCHDGV2b(MY, CY) As Double
1318
1319        Dim emisOpVOCLDDV(MY, CY) As Double
1320        Dim emisOpVOCLDDV1(MY, CY) As Double
1321        Dim emisOpVOCLDDV2(MY, CY) As Double
1322        Dim emisOpVOCHDDV2b(MY, CY) As Double
1323
1324        For i = 0 To CY
1325            For j = 0 To MY
1326                If i >= j Then
1327                    emisOpVOCLDGV(j, i) = (VMTLDGV(j, i) * emisRateVOCLDGV(j, i)) * (10 ^ 6)
     / (454 * 2000 * 1000)
1328                    emisOpVOCLDGT1(j, i) = (VMTLDGT1(j, i) * emisRateVOCLDGT1(j, i)) * (10 ^
     6) / (454 * 2000 * 1000)
1329                    emisOpVOCLDGT2(j, i) = (VMTLDGT2(j, i) * emisRateVOCLDGT2(j, i)) * (10 ^
     6) / (454 * 2000 * 1000)
1330                    emisOpVOCHDGV2b(j, i) = (VMTHDGV2b(j, i) * emisRateVOCHDGV2b(j, i)) *
    (10 ^ 6) / (454 * 2000 * 1000)
1331
1332                    emisOpVOCLDDV(j, i) = (VMTLDDV(j, i) * CAFE.Parameters.vocDiesLDDV(j,
    Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1333                    emisOpVOCLDDV1(j, i) = (VMTLDDV1(j, i) * CAFE.Parameters.vocDiesLDDT1(j,
     Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1334                    emisOpVOCLDDV2(j, i) = (VMTLDDV2(j, i) * CAFE.Parameters.vocDiesLDDT2(j,
     Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1335                    emisOpVOCHDDV2b(j, i) = (VMTHDDV2b(j, i) * CAFE.Parameters.vocDiesHDDV2b
    (j, Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1336                End If
1337            Next
```

```
1338        Next
1339
1340        'Total ("Well to Wheel") Volatile Organic Compounds (VOC) Emissions
1341        Dim emisTotVOCLDGV(MY, CY) As Double
1342        Dim emisTotVOCLDGT1(MY, CY) As Double
1343        Dim emisTotVOCLDGT2(MY, CY) As Double
1344        Dim emisTotVOCHDGV2b(MY, CY) As Double
1345
1346        Dim emisTotVOCLDDV(MY, CY) As Double
1347        Dim emisTotVOCLDDV1(MY, CY) As Double
1348        Dim emisTotVOCLDDV2(MY, CY) As Double
1349        Dim emisTotVOCHDDV2b(MY, CY) As Double
1350
1351        'TOTALS:
1352        Dim gasEmisTotVOCTotal(MY, CY) As Double
1353        Dim diesEmisTotVOCTotal(MY, CY) As Double
1354
1355        For i = 0 To CY
1356            For j = 0 To MY
1357                If i >= j Then
1358                    emisTotVOCLDGV(j, i) = emisExtrVOCLDGV(j, i) + emisTransVOCLDGV(j, i) + ↙
    emisRefVOCLDGV(j, i) + emisTSDVOCLDGV(j, i) + emisOpVOCLDGV(j, i)
1359                    emisTotVOCLDGT1(j, i) = emisExtrVOCLDGT1(j, i) + emisTransVOCLDGT1(j, i) ↙
     + emisRefVOCLDGT1(j, i) + emisTSDVOCLDGT1(j, i) + emisOpVOCLDGT1(j, i)
1360                    emisTotVOCLDGT2(j, i) = emisExtrVOCLDGT2(j, i) + emisTransVOCLDGT2(j, i) ↙
     + emisRefVOCLDGT2(j, i) + emisTSDVOCLDGT2(j, i) + emisOpVOCLDGT2(j, i)
1361                    emisTotVOCHDGV2b(j, i) = emisExtrVOCHDGV2b(j, i) + emisTransVOCHDGV2b(j, ↙
     i) + emisRefVOCHDGV2b(j, i) + emisTSDVOCHDGV2b(j, i) + emisOpVOCHDGV2b(j, i)
1362
1363                    emisTotVOCLDDV(j, i) = emisExtrVOCLDDV(j, i) + emisTransVOCLDDV(j, i) + ↙
    emisRefVOCLDDV(j, i) + emisTSDVOCLDDV(j, i) + emisOpVOCLDDV(j, i)
1364                    emisTotVOCLDDV1(j, i) = emisExtrVOCLDDV1(j, i) + emisTransVOCLDDV1(j, i) ↙
     + emisRefVOCLDDV1(j, i) + emisTSDVOCLDDV1(j, i) + emisOpVOCLDDV1(j, i)
1365                    emisTotVOCLDDV2(j, i) = emisExtrVOCLDDV2(j, i) + emisTransVOCLDDV2(j, i) ↙
     + emisRefVOCLDDV2(j, i) + emisTSDVOCLDDV2(j, i) + emisOpVOCLDDV2(j, i)
1366                    emisTotVOCHDDV2b(j, i) = emisExtrVOCHDDV2b(j, i) + emisTransVOCHDDV2b(j, ↙
     i) + emisRefVOCHDDV2b(j, i) + emisTSDVOCHDDV2b(j, i) + emisOpVOCHDDV2b(j, i)
1367
1368                    'TOTALS:
1369                    gasEmisTotVOCTotal(j, i) = emisTotVOCLDGV(j, i) + emisTotVOCLDGT1(j, i) ↙
    + emisTotVOCLDGT2(j, i) + emisTotVOCHDGV2b(j, i)
1370                    diesEmisTotVOCTotal(j, i) = emisTotVOCLDDV(j, i) + emisTotVOCLDDV1(j, i) ↙
     + emisTotVOCLDDV2(j, i) + emisTotVOCHDDV2b(j, i)
1371                End If
1372            Next
1373        Next
1374
1375        'Economic Benefit/Cost of Total Volatile Organic Compounds (VOC) Emissions
1376        Dim damageEmisTotVOCLDGV(MY, CY) As Double
1377        Dim damageEmisTotVOCLDGT1(MY, CY) As Double
1378        Dim damageEmisTotVOCLDGT2(MY, CY) As Double
1379        Dim damageEmisTotVOCHDGV2b(MY, CY) As Double
1380
1381        Dim damageEmisTotVOCLDDV(MY, CY) As Double
1382        Dim damageEmisTotVOCLDDV1(MY, CY) As Double
1383        Dim damageEmisTotVOCLDDV2(MY, CY) As Double
1384        Dim damageEmisTotVOCHDDV2b(MY, CY) As Double
1385
1386        Dim discDamageEmisTotVOCLDGV(MY, CY) As Double
1387        Dim discDamageEmisTotVOCLDGT1(MY, CY) As Double
1388        Dim discDamageEmisTotVOCLDGT2(MY, CY) As Double
1389        Dim discDamageEmisTotVOCHDGV2b(MY, CY) As Double
1390
1391        Dim discDamageEmisTotVOCLDDV(MY, CY) As Double
1392        Dim discDamageEmisTotVOCLDDV1(MY, CY) As Double
1393        Dim discDamageEmisTotVOCLDDV2(MY, CY) As Double
1394        Dim discDamageEmisTotVOCHDDV2b(MY, CY) As Double
1395
1396        'TOTALS:
```

```
        Dim gasDamageEmisTotVOCTotal(MY, CY) As Double
        Dim diesDamageEmisTotVOCTotal(MY, CY) As Double

        Dim discGasDamageEmisTotVOCTotal(MY, CY) As Double
        Dim discDiesDamageEmisTotVOCTotal(MY, CY) As Double

        For i = 0 To CY
            For j = 0 To MY
                If i >= j Then
                        damageEmisTotVOCLDGV(j, i) = emisTotVOCLDGV(j, i) * CAFE.Parameters.damageCostVOC / 1000
                        damageEmisTotVOCLDGT1(j, i) = emisTotVOCLDGT1(j, i) * CAFE.Parameters.damageCostVOC / 1000
                        damageEmisTotVOCLDGT2(j, i) = emisTotVOCLDGT2(j, i) * CAFE.Parameters.damageCostVOC / 1000
                        damageEmisTotVOCHDGV2b(j, i) = emisTotVOCHDGV2b(j, i) * CAFE.Parameters.damageCostVOC / 1000

                        damageEmisTotVOCLDDV(j, i) = emisTotVOCLDDV(j, i) * CAFE.Parameters.damageCostVOC / 1000
                        damageEmisTotVOCLDDV1(j, i) = emisTotVOCLDDV1(j, i) * CAFE.Parameters.damageCostVOC / 1000
                        damageEmisTotVOCLDDV2(j, i) = emisTotVOCLDDV2(j, i) * CAFE.Parameters.damageCostVOC / 1000
                        damageEmisTotVOCHDDV2b(j, i) = emisTotVOCHDDV2b(j, i) * CAFE.Parameters.damageCostVOC / 1000

                        discDamageEmisTotVOCLDGV(j, i) = damageEmisTotVOCLDGV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotVOCLDGT1(j, i) = damageEmisTotVOCLDGT1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotVOCLDGT2(j, i) = damageEmisTotVOCLDGT2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotVOCHDGV2b(j, i) = damageEmisTotVOCHDGV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                        discDamageEmisTotVOCLDDV(j, i) = damageEmisTotVOCLDDV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotVOCLDDV1(j, i) = damageEmisTotVOCLDDV1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotVOCLDDV2(j, i) = damageEmisTotVOCLDDV2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotVOCHDDV2b(j, i) = damageEmisTotVOCHDDV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                        'TOTALS:
                        gasDamageEmisTotVOCTotal(j, i) = damageEmisTotVOCLDGV(j, i) + damageEmisTotVOCLDGT1(j, i) + damageEmisTotVOCLDGT2(j, i) + damageEmisTotVOCHDGV2b(j, i)
                        diesDamageEmisTotVOCTotal(j, i) = damageEmisTotVOCLDDV(j, i) + damageEmisTotVOCLDDV1(j, i) + damageEmisTotVOCLDDV2(j, i) + damageEmisTotVOCHDDV2b(j, i)

                        discGasDamageEmisTotVOCTotal(j, i) = discDamageEmisTotVOCLDGV(j, i) + discDamageEmisTotVOCLDGT1(j, i) + discDamageEmisTotVOCLDGT2(j, i) + discDamageEmisTotVOCHDGV2b(j, i)
                        discDiesDamageEmisTotVOCTotal(j, i) = discDamageEmisTotVOCLDDV(j, i) + discDamageEmisTotVOCLDDV1(j, i) + discDamageEmisTotVOCLDDV2(j, i) + discDamageEmisTotVOCHDDV2b(j, i)

                End If
            Next
        Next




        '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        'NITROUS OXIDES (NOX)
        'average NOX upstream emissions (grams/MBTU) vars:
```

```
1443        Dim avgUpEmisExtrNOX As Double
1444        Dim avgUpEmisTransNOX As Double
1445        Dim avgUpEmisRefNOX As Double
1446        Dim avgUpEmisTSDNOX As Double
1447
1448        Dim avgUpEmisExtrDiesNOX As Double
1449        Dim avgUpEmisTransDiesNOX As Double
1450        Dim avgUpEmisRefDiesNOX As Double
1451        Dim avgUpEmisTSDDiesNOX As Double
1452
1453        'calculate NOX average upstream emissions (grams/MBTU)
1454        avgUpEmisExtrNOX = (CAFE.Parameters.totExtrConvGasNOX * CAFE.Parameters.convGasMix) ↙
        + (CAFE.Parameters.totExtrFedGasNOX * CAFE.Parameters.fedGasMix) + (CAFE.Parameters. ↙
        totExtrCaliGasNOX * CAFE.Parameters.caliGasMix)
1455        avgUpEmisTransNOX = (CAFE.Parameters.totTransConvGasNOX * CAFE.Parameters. ↙
        convGasMix) + (CAFE.Parameters.totTransFedGasNOX * CAFE.Parameters.fedGasMix) + (CAFE. ↙
        Parameters.totTransCaliGasNOX * CAFE.Parameters.caliGasMix)
1456        avgUpEmisRefNOX = (CAFE.Parameters.totRefConvGasNOX * CAFE.Parameters.convGasMix) + ↙
        (CAFE.Parameters.totRefFedGasNOX * CAFE.Parameters.fedGasMix) + (CAFE.Parameters. ↙
        totRefCaliGasNOX * CAFE.Parameters.caliGasMix)
1457        avgUpEmisTSDNOX = (CAFE.Parameters.totTSDConvGasNOX * CAFE.Parameters.convGasMix) + ↙
        (CAFE.Parameters.totTSDFedGasNOX * CAFE.Parameters.fedGasMix) + (CAFE.Parameters. ↙
        totTSDCaliGasNOX * CAFE.Parameters.caliGasMix)
1458
1459        avgUpEmisExtrDiesNOX = (CAFE.Parameters.totExtrConvDiesNOX * CAFE.Parameters. ↙
        convDiesMix) + (CAFE.Parameters.totExtrLowDiesNOX * CAFE.Parameters.lowDiesMix)
1460        avgUpEmisTransDiesNOX = (CAFE.Parameters.totTransConvDiesNOX * CAFE.Parameters. ↙
        convDiesMix) + (CAFE.Parameters.totTransLowDiesNOX * CAFE.Parameters.lowDiesMix)
1461        avgUpEmisRefDiesNOX = (CAFE.Parameters.totRefConvDiesNOX * CAFE.Parameters. ↙
        convDiesMix) + (CAFE.Parameters.totRefLowDiesNOX * CAFE.Parameters.lowDiesMix)
1462        avgUpEmisTSDDiesNOX = (CAFE.Parameters.totTSDConvDiesNOX * CAFE.Parameters. ↙
        convDiesMix) + (CAFE.Parameters.totTSDLowDiesNOX * CAFE.Parameters.lowDiesMix)
1463
1464        'adjusted NOX average upstream emissions (grams/gal) vars:
1465        Dim adjUpEmisExtrNOX As Double
1466        Dim adjUpEmisTransNOX As Double
1467        Dim adjUpEmisRefNOX As Double
1468        Dim adjUpEmisTSDNOX As Double
1469
1470        Dim adjUpEmisExtrDiesNOX As Double
1471        Dim adjUpEmisTransDiesNOX As Double
1472        Dim adjUpEmisRefDiesNOX As Double
1473        Dim adjUpEmisTSDDiesNOX As Double
1474
1475        'calculate NOX adjusted average upstream emissions (grams/gal):
1476        adjUpEmisExtrNOX = avgUpEmisExtrNOX * (avgGasEng / 10 ^ 6)
1477        adjUpEmisTransNOX = avgUpEmisTransNOX * (avgGasEng / 10 ^ 6)
1478        adjUpEmisRefNOX = avgUpEmisRefNOX * (avgGasEng / 10 ^ 6)
1479        adjUpEmisTSDNOX = avgUpEmisTSDNOX * (avgGasEng / 10 ^ 6)
1480
1481        adjUpEmisExtrDiesNOX = avgUpEmisExtrDiesNOX * (avgDiesEng / 10 ^ 6)
1482        adjUpEmisTransDiesNOX = avgUpEmisTransDiesNOX * (avgDiesEng / 10 ^ 6)
1483        adjUpEmisRefDiesNOX = avgUpEmisRefDiesNOX * (avgDiesEng / 10 ^ 6)
1484        adjUpEmisTSDDiesNOX = avgUpEmisTSDDiesNOX * (avgDiesEng / 10 ^ 6)
1485
1486        'Nitrous Oxide  (NOX) Emissions from Domestic Crude Petroleum Extraction
1487        Dim emisExtrNOXLDGV(MY, CY) As Double
1488        Dim emisExtrNOXLDGT1(MY, CY) As Double
1489        Dim emisExtrNOXLDGT2(MY, CY) As Double
1490        Dim emisExtrNOXHDGV2b(MY, CY) As Double
1491
1492        Dim emisExtrNOXLDDV(MY, CY) As Double
1493        Dim emisExtrNOXLDDV1(MY, CY) As Double
1494        Dim emisExtrNOXLDDV2(MY, CY) As Double
1495        Dim emisExtrNOXHDDV2b(MY, CY) As Double
1496
1497        For i = 0 To CY
1498            For j = 0 To MY
1499                If i >= j Then
```

```
1500                    emisExtrNOXLDGV(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
      gasLowDomRefDomCrude) * (adjUpEmisExtrNOX * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 *
      1000)
1501                    emisExtrNOXLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
      .gasLowDomRefDomCrude) * (adjUpEmisExtrNOX * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 *
      1000)
1502                    emisExtrNOXLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
      .gasLowDomRefDomCrude) * (adjUpEmisExtrNOX * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 *
      1000)
1503                    emisExtrNOXHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.
      Parameters.gasLowDomRefDomCrude) * (adjUpEmisExtrNOX * galHDGV2b(j, i) * 10 ^ 6) / (454
      * 2000 * 1000)
1504
1505                    emisExtrNOXLDDV(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters
      .diesLowDomRefDomCrude) * (adjUpEmisExtrDiesNOX * galLDDV(j, i) * 10 ^ 6) / (454 * 2000
      * 1000)
1506                    emisExtrNOXLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
      Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesNOX * galLDDV1(j, i) * 10 ^ 6) /
      (454 * 2000 * 1000)
1507                    emisExtrNOXLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
      Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesNOX * galLDDV2(j, i) * 10 ^ 6) /
      (454 * 2000 * 1000)
1508                    emisExtrNOXHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
      Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesNOX * galHDDV2b(j, i) * 10 ^ 6) /
      (454 * 2000 * 1000)
1509                End If
1510            Next
1511        Next
1512
1513        'Nitrous Oxide (NOX) Emissions from Domestic Crude Petroleum Transportation
1514        Dim emisTransNOXLDGV(MY, CY) As Double
1515        Dim emisTransNOXLDGT1(MY, CY) As Double
1516        Dim emisTransNOXLDGT2(MY, CY) As Double
1517        Dim emisTransNOXHDGV2b(MY, CY) As Double
1518
1519        Dim emisTransNOXLDDV(MY, CY) As Double
1520        Dim emisTransNOXLDDV1(MY, CY) As Double
1521        Dim emisTransNOXLDDV2(MY, CY) As Double
1522        Dim emisTransNOXHDDV2b(MY, CY) As Double
1523
1524        For i = 0 To CY
1525            For j = 0 To MY
1526                If i >= j Then
1527                    emisTransNOXLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) *
      (adjUpEmisTransNOX * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1528                    emisTransNOXLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) *
      (adjUpEmisTransNOX * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1529                    emisTransNOXLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) *
      (adjUpEmisTransNOX * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1530                    emisTransNOXHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) *
      (adjUpEmisTransNOX * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1531
1532                    emisTransNOXLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
      (adjUpEmisTransDiesNOX * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1533                    emisTransNOXLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
      (adjUpEmisTransDiesNOX * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1534                    emisTransNOXLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
      (adjUpEmisTransDiesNOX * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1535                    emisTransNOXHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
      (adjUpEmisTransDiesNOX * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1536                End If
1537            Next
1538        Next
1539
1540        ' Nitrous Oxide  (NOX) Emissions from Domestic Crude Petroleum Refining to Produce
      Gasoline
1541        Dim emisRefNOXLDGV(MY, CY) As Double
1542        Dim emisRefNOXLDGT1(MY, CY) As Double
1543        Dim emisRefNOXLDGT2(MY, CY) As Double
```

```
1544        Dim emisRefNOXHDGV2b(MY, CY) As Double
1545
1546        Dim emisRefNOXLDDV(MY, CY) As Double
1547        Dim emisRefNOXLDDV1(MY, CY) As Double
1548        Dim emisRefNOXLDDV2(MY, CY) As Double
1549        Dim emisRefNOXHDDV2b(MY, CY) As Double
1550
1551        For i = 0 To CY
1552            For j = 0 To MY
1553                If i >= j Then
1554                    emisRefNOXLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefNOX
     * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1555                    emisRefNOXLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisRefNOX * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1556                    emisRefNOXLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisRefNOX * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1557                    emisRefNOXHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisRefNOX * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1558
1559                    emisRefNOXLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesNOX * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1560                    emisRefNOXLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesNOX * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1561                    emisRefNOXLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesNOX * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1562                    emisRefNOXHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesNOX * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1563                End If
1564            Next
1565        Next
1566
1567        'Nitrous Oxide  (NOX) Emissions from Gasoline Transportation, Storage, and
   Distribution
1568        Dim emisTSDNOXLDGV(MY, CY) As Double
1569        Dim emisTSDNOXLDGT1(MY, CY) As Double
1570        Dim emisTSDNOXLDGT2(MY, CY) As Double
1571        Dim emisTSDNOXHDGV2b(MY, CY) As Double
1572
1573        Dim emisTSDNOXLDDV(MY, CY) As Double
1574        Dim emisTSDNOXLDDV1(MY, CY) As Double
1575        Dim emisTSDNOXLDDV2(MY, CY) As Double
1576        Dim emisTSDNOXHDDV2b(MY, CY) As Double
1577
1578        For i = 0 To CY
1579            For j = 0 To MY
1580                If i >= j Then
1581                    emisTSDNOXLDGV(j, i) = (adjUpEmisTSDNOX * galLDGV(j, i) * 10 ^ 6) / (454
     * 2000 * 1000)
1582                    emisTSDNOXLDGT1(j, i) = (adjUpEmisTSDNOX * galLDGT1(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1583                    emisTSDNOXLDGT2(j, i) = (adjUpEmisTSDNOX * galLDGT2(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1584                    emisTSDNOXHDGV2b(j, i) = (adjUpEmisTSDNOX * galHDGV2b(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1585
1586                    emisTSDNOXLDDV(j, i) = (adjUpEmisTSDDiesNOX * galLDDV(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1587                    emisTSDNOXLDDV1(j, i) = (adjUpEmisTSDDiesNOX * galLDDV1(j, i) * 10 ^ 6)
   / (454 * 2000 * 1000)
1588                    emisTSDNOXLDDV2(j, i) = (adjUpEmisTSDDiesNOX * galLDDV2(j, i) * 10 ^ 6)
   / (454 * 2000 * 1000)
1589                    emisTSDNOXHDDV2b(j, i) = (adjUpEmisTSDDiesNOX * galHDDV2b(j, i) * 10 ^
   6) / (454 * 2000 * 1000)
1590                End If
1591            Next
1592        Next
1593
1594
1595        ''Nitrous Oxide (NOX) Emissions from Vehicle Operation ("Tank to Wheel")
```

```
1958                    emisTotPMHDGV2b(j, i) = emisExtrPMHDGV2b(j, i) + emisTransPMHDGV2b(j, i) + emisRefPMHDGV2b(j, i) + emisTSDPMHDGV2b(j, i) + emisOpPMHDGV2b(j, i)
1959
1960                    emisTotPMLDDV(j, i) = emisExtrPMLDDV(j, i) + emisTransPMLDDV(j, i) + emisRefPMLDDV(j, i) + emisTSDPMLDDV(j, i) + emisOpPMLDDV(j, i)
1961                    emisTotPMLDDV1(j, i) = emisExtrPMLDDV1(j, i) + emisTransPMLDDV1(j, i) + emisRefPMLDDV1(j, i) + emisTSDPMLDDV1(j, i) + emisOpPMLDDV1(j, i)
1962                    emisTotPMLDDV2(j, i) = emisExtrPMLDDV2(j, i) + emisTransPMLDDV2(j, i) + emisRefPMLDDV2(j, i) + emisTSDPMLDDV2(j, i) + emisOpPMLDDV2(j, i)
1963                    emisTotPMHDDV2b(j, i) = emisExtrPMHDDV2b(j, i) + emisTransPMHDDV2b(j, i) + emisRefPMHDDV2b(j, i) + emisTSDPMHDDV2b(j, i) + emisOpPMHDDV2b(j, i)
1964
1965                    'TOTALS:
1966                    gasEmisTotPMTotal(j, i) = emisTotPMLDGV(j, i) + emisTotPMLDGT1(j, i) + emisTotPMLDGT2(j, i) + emisTotPMHDGV2b(j, i)
1967                    diesEmisTotPMTotal(j, i) = emisTotPMLDDV(j, i) + emisTotPMLDDV1(j, i) + emisTotPMLDDV2(j, i) + emisTotPMHDDV2b(j, i)
1968                End If
1969            Next
1970        Next
1971
1972        'Economic Benefit/Cost of Total Particulate Matter (PM) Emissions
1973        Dim damageEmisTotPMLDGV(MY, CY) As Double
1974        Dim damageEmisTotPMLDGT1(MY, CY) As Double
1975        Dim damageEmisTotPMLDGT2(MY, CY) As Double
1976        Dim damageEmisTotPMHDGV2b(MY, CY) As Double
1977
1978        Dim damageEmisTotPMLDDV(MY, CY) As Double
1979        Dim damageEmisTotPMLDDV1(MY, CY) As Double
1980        Dim damageEmisTotPMLDDV2(MY, CY) As Double
1981        Dim damageEmisTotPMHDDV2b(MY, CY) As Double
1982
1983        Dim discDamageEmisTotPMLDGV(MY, CY) As Double
1984        Dim discDamageEmisTotPMLDGT1(MY, CY) As Double
1985        Dim discDamageEmisTotPMLDGT2(MY, CY) As Double
1986        Dim discDamageEmisTotPMHDGV2b(MY, CY) As Double
1987
1988        Dim discDamageEmisTotPMLDDV(MY, CY) As Double
1989        Dim discDamageEmisTotPMLDDV1(MY, CY) As Double
1990        Dim discDamageEmisTotPMLDDV2(MY, CY) As Double
1991        Dim discDamageEmisTotPMHDDV2b(MY, CY) As Double
1992
1993        'TOTALS:
1994        Dim gasDamageEmisTotPMTotal(MY, CY) As Double
1995        Dim diesDamageEmisTotPMTotal(MY, CY) As Double
1996
1997        Dim discGasDamageEmisTotPMTotal(MY, CY) As Double
1998        Dim discDiesDamageEmisTotPMTotal(MY, CY) As Double
1999
2000        For i = 0 To CY
2001            For j = 0 To MY
2002                If i >= j Then
2003                    damageEmisTotPMLDGV(j, i) = emisTotPMLDGV(j, i) * CAFE.Parameters.damageCostPM / 1000
2004                    damageEmisTotPMLDGT1(j, i) = emisTotPMLDGT1(j, i) * CAFE.Parameters.damageCostPM / 1000
2005                    damageEmisTotPMLDGT2(j, i) = emisTotPMLDGT2(j, i) * CAFE.Parameters.damageCostPM / 1000
2006                    damageEmisTotPMHDGV2b(j, i) = emisTotPMHDGV2b(j, i) * CAFE.Parameters.damageCostPM / 1000
2007
2008                    damageEmisTotPMLDDV(j, i) = emisTotPMLDDV(j, i) * CAFE.Parameters.damageCostPM / 1000
2009                    damageEmisTotPMLDDV1(j, i) = emisTotPMLDDV1(j, i) * CAFE.Parameters.damageCostPM / 1000
2010                    damageEmisTotPMLDDV2(j, i) = emisTotPMLDDV2(j, i) * CAFE.Parameters.damageCostPM / 1000
2011                    damageEmisTotPMHDDV2b(j, i) = emisTotPMHDDV2b(j, i) * CAFE.Parameters.damageCostPM / 1000
```

```

                        discDamageEmisTotPMLDGV(j, i) = damageEmisTotPMLDGV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotPMLDGT1(j, i) = damageEmisTotPMLDGT1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotPMLDGT2(j, i) = damageEmisTotPMLDGT2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotPMHDGV2b(j, i) = damageEmisTotPMHDGV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                        discDamageEmisTotPMLDDV(j, i) = damageEmisTotPMLDDV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotPMLDDV1(j, i) = damageEmisTotPMLDDV1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotPMLDDV2(j, i) = damageEmisTotPMLDDV2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotPMHDDV2b(j, i) = damageEmisTotPMHDDV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                        'TOTALS:
                        gasDamageEmisTotPMTotal(j, i) = damageEmisTotPMLDGV(j, i) + damageEmisTotPMLDGT1(j, i) + damageEmisTotPMLDGT2(j, i) + damageEmisTotPMHDGV2b(j, i)
                        diesDamageEmisTotPMTotal(j, i) = damageEmisTotPMLDDV(j, i) + damageEmisTotPMLDDV1(j, i) + damageEmisTotPMLDDV2(j, i) + damageEmisTotPMHDDV2b(j, i)

                        discGasDamageEmisTotPMTotal(j, i) = discDamageEmisTotPMLDGV(j, i) + discDamageEmisTotPMLDGT1(j, i) + discDamageEmisTotPMLDGT2(j, i) + discDamageEmisTotPMHDGV2b(j, i)
                        discDiesDamageEmisTotPMTotal(j, i) = discDamageEmisTotPMLDDV(j, i) + discDamageEmisTotPMLDDV1(j, i) + discDamageEmisTotPMLDDV2(j, i) + discDamageEmisTotPMHDDV2b(j, i)

                    End If
                Next
            Next




            ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
            'SULFUR DIOXIDE (SO2)
            'average SO2 upstream emissions (grams/MBTU) vars:
            Dim avgUpEmisExtrSOX As Double
            Dim avgUpEmisTransSOX As Double
            Dim avgUpEmisRefSOX As Double
            Dim avgUpEmisTSDSOX As Double

            Dim avgUpEmisExtrDiesSOX As Double
            Dim avgUpEmisTransDiesSOX As Double
            Dim avgUpEmisRefDiesSOX As Double
            Dim avgUpEmisTSDDiesSOX As Double

            'calculate SO2 average upstream emissions (grams/MBTU)
            avgUpEmisExtrSOX = (CAFE.Parameters.totExtrConvGasSOX * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totExtrFedGasSOX * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totExtrCaliGasSOX * CAFE.Parameters.caliGasMix)
            avgUpEmisTransSOX = (CAFE.Parameters.totTransConvGasSOX * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totTransFedGasSOX * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totTransCaliGasSOX * CAFE.Parameters.caliGasMix)
            avgUpEmisRefSOX = (CAFE.Parameters.totRefConvGasSOX * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totRefFedGasSOX * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totRefCaliGasSOX * CAFE.Parameters.caliGasMix)
            avgUpEmisTSDSOX = (CAFE.Parameters.totTSDConvGasSOX * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totTSDFedGasSOX * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.totTSDCaliGasSOX * CAFE.Parameters.caliGasMix)

            avgUpEmisExtrDiesSOX = (CAFE.Parameters.totExtrConvDiesSOX * CAFE.Parameters.convDiesMix) + (CAFE.Parameters.totExtrLowDiesSOX * CAFE.Parameters.lowDiesMix)
```

```
2057        avgUpEmisTransDiesSOX = (CAFE.Parameters.totTransConvDiesSOX * CAFE.Parameters.
        convDiesMix) + (CAFE.Parameters.totTransLowDiesSOX * CAFE.Parameters.lowDiesMix)
2058        avgUpEmisRefDiesSOX = (CAFE.Parameters.totRefConvDiesSOX * CAFE.Parameters.
        convDiesMix) + (CAFE.Parameters.totRefLowDiesSOX * CAFE.Parameters.lowDiesMix)
2059        avgUpEmisTSDDiesSOX = (CAFE.Parameters.totTSDConvDiesSOX * CAFE.Parameters.
        convDiesMix) + (CAFE.Parameters.totTSDLowDiesSOX * CAFE.Parameters.lowDiesMix)
2060
2061        'adjusted SO2 average upstream emissions (grams/gal) vars:
2062        Dim adjUpEmisExtrSOX As Double
2063        Dim adjUpEmisTransSOX As Double
2064        Dim adjUpEmisRefSOX As Double
2065        Dim adjUpEmisTSDSOX As Double
2066
2067        Dim adjUpEmisExtrDiesSOX As Double
2068        Dim adjUpEmisTransDiesSOX As Double
2069        Dim adjUpEmisRefDiesSOX As Double
2070        Dim adjUpEmisTSDDiesSOX As Double
2071
2072        'calculate SO2 adjusted average upstream emissions (grams/gal):
2073        adjUpEmisExtrSOX = avgUpEmisExtrSOX * (avgGasEng / 10 ^ 6)
2074        adjUpEmisTransSOX = avgUpEmisTransSOX * (avgGasEng / 10 ^ 6)
2075        adjUpEmisRefSOX = avgUpEmisRefSOX * (avgGasEng / 10 ^ 6)
2076        adjUpEmisTSDSOX = avgUpEmisTSDSOX * (avgGasEng / 10 ^ 6)
2077
2078        adjUpEmisExtrDiesSOX = avgUpEmisExtrDiesSOX * (avgDiesEng / 10 ^ 6)
2079        adjUpEmisTransDiesSOX = avgUpEmisTransDiesSOX * (avgDiesEng / 10 ^ 6)
2080        adjUpEmisRefDiesSOX = avgUpEmisRefDiesSOX * (avgDiesEng / 10 ^ 6)
2081        adjUpEmisTSDDiesSOX = avgUpEmisTSDDiesSOX * (avgDiesEng / 10 ^ 6)
2082
2083        'Sulfur Dioxide (SO2) Emissions from Domestic Crude Petroleum Extraction
2084        Dim emisExtrSOXLDGV(MY, CY) As Double
2085        Dim emisExtrSOXLDGT1(MY, CY) As Double
2086        Dim emisExtrSOXLDGT2(MY, CY) As Double
2087        Dim emisExtrSOXHDGV2b(MY, CY) As Double
2088
2089        Dim emisExtrSOXLDDV(MY, CY) As Double
2090        Dim emisExtrSOXLDDV1(MY, CY) As Double
2091        Dim emisExtrSOXLDDV2(MY, CY) As Double
2092        Dim emisExtrSOXHDDV2b(MY, CY) As Double
2093
2094        For i = 0 To CY
2095            For j = 0 To MY
2096                If i >= j Then
2097                    emisExtrSOXLDGV(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
        gasLowDomRefDomCrude) * (adjUpEmisExtrSOX * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 *
        1000)
2098                    emisExtrSOXLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
        .gasLowDomRefDomCrude) * (adjUpEmisExtrSOX * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 *
        1000)
2099                    emisExtrSOXLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
        .gasLowDomRefDomCrude) * (adjUpEmisExtrSOX * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 *
        1000)
2100                    emisExtrSOXHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.
        Parameters.gasLowDomRefDomCrude) * (adjUpEmisExtrSOX * galHDGV2b(j, i) * 10 ^ 6) / (454
        * 2000 * 1000)
2101
2102                    emisExtrSOXLDDV(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters
        .diesLowDomRefDomCrude) * (adjUpEmisExtrDiesSOX * galLDDV(j, i) * 10 ^ 6) / (454 * 2000
        * 1000)
2103                    emisExtrSOXLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
        Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesSOX * galLDDV1(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
2104                    emisExtrSOXLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
        Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesSOX * galLDDV2(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
2105                    emisExtrSOXHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.
        Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesSOX * galHDDV2b(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
2106                End If
```

```
2107            Next
2108        Next
2109
2110        'Sulfur Dioxide (SO2) Emissions from Domestic Crude Petroleum Transportation
2111        Dim emisTransSOXLDGV(MY, CY) As Double
2112        Dim emisTransSOXLDGT1(MY, CY) As Double
2113        Dim emisTransSOXLDGT2(MY, CY) As Double
2114        Dim emisTransSOXHDGV2b(MY, CY) As Double
2115
2116        Dim emisTransSOXLDDV(MY, CY) As Double
2117        Dim emisTransSOXLDDV1(MY, CY) As Double
2118        Dim emisTransSOXLDDV2(MY, CY) As Double
2119        Dim emisTransSOXHDDV2b(MY, CY) As Double
2120
2121        For i = 0 To CY
2122            For j = 0 To MY
2123                If i >= j Then
2124                    emisTransSOXLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransSOX * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2125                    emisTransSOXLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransSOX * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2126                    emisTransSOXLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransSOX * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2127                    emisTransSOXHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisTransSOX * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2128
2129                    emisTransSOXLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesSOX * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2130                    emisTransSOXLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesSOX * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2131                    emisTransSOXLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesSOX * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2132                    emisTransSOXHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisTransDiesSOX * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2133                End If
2134            Next
2135        Next
2136
2137        ' Sulfur Dioxide (SO2) Emissions from Domestic Crude Petroleum Refining to Produce
    Gasoline
2138        Dim emisRefSOXLDGV(MY, CY) As Double
2139        Dim emisRefSOXLDGT1(MY, CY) As Double
2140        Dim emisRefSOXLDGT2(MY, CY) As Double
2141        Dim emisRefSOXHDGV2b(MY, CY) As Double
2142
2143        Dim emisRefSOXLDDV(MY, CY) As Double
2144        Dim emisRefSOXLDDV1(MY, CY) As Double
2145        Dim emisRefSOXLDDV2(MY, CY) As Double
2146        Dim emisRefSOXHDDV2b(MY, CY) As Double
2147
2148        For i = 0 To CY
2149            For j = 0 To MY
2150                If i >= j Then
2151                    emisRefSOXLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefSOX
     * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2152                    emisRefSOXLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisRefSOX * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2153                    emisRefSOXLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisRefSOX * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2154                    emisRefSOXHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) *
    (adjUpEmisRefSOX * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2155
2156                    emisRefSOXLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesSOX * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2157                    emisRefSOXLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesSOX * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2158                    emisRefSOXLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesSOX * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2159                    emisRefSOXHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
```

```
        (adjUpEmisRefDiesSOX * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
2160                End If
2161            Next
2162        Next
2163
2164        'Sulfur Dioxide (SO2) Emissions from Gasoline Transportation, Storage, and
        Distribution
2165        Dim emisTSDSOXLDGV(MY, CY) As Double
2166        Dim emisTSDSOXLDGT1(MY, CY) As Double
2167        Dim emisTSDSOXLDGT2(MY, CY) As Double
2168        Dim emisTSDSOXHDGV2b(MY, CY) As Double
2169
2170        Dim emisTSDSOXLDDV(MY, CY) As Double
2171        Dim emisTSDSOXLDDV1(MY, CY) As Double
2172        Dim emisTSDSOXLDDV2(MY, CY) As Double
2173        Dim emisTSDSOXHDDV2b(MY, CY) As Double
2174
2175        For i = 0 To CY
2176            For j = 0 To MY
2177                If i >= j Then
2178                    emisTSDSOXLDGV(j, i) = (adjUpEmisTSDSOX * galLDGV(j, i) * 10 ^ 6) / (454
         * 2000 * 1000)
2179                    emisTSDSOXLDGT1(j, i) = (adjUpEmisTSDSOX * galLDGT1(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
2180                    emisTSDSOXLDGT2(j, i) = (adjUpEmisTSDSOX * galLDGT2(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
2181                    emisTSDSOXHDGV2b(j, i) = (adjUpEmisTSDSOX * galHDGV2b(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
2182
2183                    emisTSDSOXLDDV(j, i) = (adjUpEmisTSDDiesSOX * galLDDV(j, i) * 10 ^ 6) /
        (454 * 2000 * 1000)
2184                    emisTSDSOXLDDV1(j, i) = (adjUpEmisTSDDiesSOX * galLDDV1(j, i) * 10 ^ 6)
        / (454 * 2000 * 1000)
2185                    emisTSDSOXLDDV2(j, i) = (adjUpEmisTSDDiesSOX * galLDDV2(j, i) * 10 ^ 6)
        / (454 * 2000 * 1000)
2186                    emisTSDSOXHDDV2b(j, i) = (adjUpEmisTSDDiesSOX * galHDDV2b(j, i) * 10 ^
        6) / (454 * 2000 * 1000)
2187                End If
2188            Next
2189        Next
2190
2191
2192        ''Sulfur Dioxide (SO2) Emissions from Vehicle Operation ("Tank to Wheel")
2193        ' calculate emission rates for gas = (Conv% * ConvRate) + ((1-Conv%) * RFGRate)
2194        Dim emisRateSO2LDGV(MY, CY) As Double
2195        Dim emisRateSO2LDGT1(MY, CY) As Double
2196        Dim emisRateSO2LDGT2(MY, CY) As Double
2197        Dim emisRateSO2HDGV2b(MY, CY) As Double
2198
2199        For i = 0 To CY
2200            For j = 0 To MY
2201                If i >= j Then
2202                    emisRateSO2LDGV(j, i) = (CAFE.Parameters.so2GasLDGV(j, Math.Min(i, 25))
        * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters.
        so2GasRFGLDGV(j, Math.Min(i, 25)))
2203                    emisRateSO2LDGT1(j, i) = (CAFE.Parameters.so2GasLDGT1(j, Math.Min(i,
        25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters
        .so2GasRFGLDGT1(j, Math.Min(i, 25)))
2204                    emisRateSO2LDGT2(j, i) = (CAFE.Parameters.so2GasLDGT2(j, Math.Min(i,
        25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters
        .so2GasRFGLDGT2(j, Math.Min(i, 25)))
2205                    emisRateSO2HDGV2b(j, i) = (CAFE.Parameters.so2GasHDGV2b(j, Math.Min(i,
        25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters
        .so2GasRFGHDGV2b(j, Math.Min(i, 25)))
2206                End If
2207            Next j
2208        Next i
2209
2210        Dim emisOpSO2LDGV(MY, CY) As Double
```

```
2211        Dim emisOpSO2LDGT1(MY, CY) As Double
2212        Dim emisOpSO2LDGT2(MY, CY) As Double
2213        Dim emisOpSO2HDGV2b(MY, CY) As Double
2214
2215        Dim emisOpSO2LDDV(MY, CY) As Double
2216        Dim emisOpSO2LDDV1(MY, CY) As Double
2217        Dim emisOpSO2LDDV2(MY, CY) As Double
2218        Dim emisOpSO2HDDV2b(MY, CY) As Double
2219
2220        For i = 0 To CY
2221            For j = 0 To MY
2222                If i >= j Then
2223                    emisOpSO2LDGV(j, i) = (VMTLDGV(j, i) * emisRateSO2LDGV(j, i)) * (10 ^ 6) / (454 * 2000 * 1000)
2224                    emisOpSO2LDGT1(j, i) = (VMTLDGT1(j, i) * emisRateSO2LDGT1(j, i)) * (10 ^ 6) / (454 * 2000 * 1000)
2225                    emisOpSO2LDGT2(j, i) = (VMTLDGT2(j, i) * emisRateSO2LDGT2(j, i)) * (10 ^ 6) / (454 * 2000 * 1000)
2226                    emisOpSO2HDGV2b(j, i) = (VMTHDGV2b(j, i) * emisRateSO2HDGV2b(j, i)) * (10 ^ 6) / (454 * 2000 * 1000)
2227
2228                    emisOpSO2LDDV(j, i) = (VMTLDDV(j, i) * CAFE.Parameters.so2DiesLDDV(j, Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
2229                    emisOpSO2LDDV1(j, i) = (VMTLDDV1(j, i) * CAFE.Parameters.so2DiesLDDT1(j, Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
2230                    emisOpSO2LDDV2(j, i) = (VMTLDDV2(j, i) * CAFE.Parameters.so2DiesLDDT2(j, Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
2231                    emisOpSO2HDDV2b(j, i) = (VMTHDDV2b(j, i) * CAFE.Parameters.so2DiesHDDV2b(j, Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
2232                End If
2233            Next
2234        Next
2235
2236        'Total ("Well to Wheel") Sulfur Dioxide (SO2) Emissions
2237        Dim emisTotSOXLDGV(MY, CY) As Double
2238        Dim emisTotSOXLDGT1(MY, CY) As Double
2239        Dim emisTotSOXLDGT2(MY, CY) As Double
2240        Dim emisTotSOXHDGV2b(MY, CY) As Double
2241
2242        Dim emisTotSOXLDDV(MY, CY) As Double
2243        Dim emisTotSOXLDDV1(MY, CY) As Double
2244        Dim emisTotSOXLDDV2(MY, CY) As Double
2245        Dim emisTotSOXHDDV2b(MY, CY) As Double
2246
2247        'TOTALS:
2248        Dim gasEmisTotSOXTotal(MY, CY) As Double
2249        Dim diesEmisTotSOXTotal(MY, CY) As Double
2250
2251        For i = 0 To CY
2252            For j = 0 To MY
2253                If i >= j Then
2254                    emisTotSOXLDGV(j, i) = emisExtrSOXLDGV(j, i) + emisTransSOXLDGV(j, i) + emisRefSOXLDGV(j, i) + emisTSDSOXLDGV(j, i) + emisOpSO2LDGV(j, i)
2255                    emisTotSOXLDGT1(j, i) = emisExtrSOXLDGT1(j, i) + emisTransSOXLDGT1(j, i) + emisRefSOXLDGT1(j, i) + emisTSDSOXLDGT1(j, i) + emisOpSO2LDGT1(j, i)
2256                    emisTotSOXLDGT2(j, i) = emisExtrSOXLDGT2(j, i) + emisTransSOXLDGT2(j, i) + emisRefSOXLDGT2(j, i) + emisTSDSOXLDGT2(j, i) + emisOpSO2LDGT2(j, i)
2257                    emisTotSOXHDGV2b(j, i) = emisExtrSOXHDGV2b(j, i) + emisTransSOXHDGV2b(j, i) + emisRefSOXHDGV2b(j, i) + emisTSDSOXHDGV2b(j, i) + emisOpSO2HDGV2b(j, i)
2258
2259                    emisTotSOXLDDV(j, i) = emisExtrSOXLDDV(j, i) + emisTransSOXLDDV(j, i) + emisRefSOXLDDV(j, i) + emisTSDSOXLDDV(j, i) + emisOpSO2LDDV(j, i)
2260                    emisTotSOXLDDV1(j, i) = emisExtrSOXLDDV1(j, i) + emisTransSOXLDDV1(j, i) + emisRefSOXLDDV1(j, i) + emisTSDSOXLDDV1(j, i) + emisOpSO2LDDV1(j, i)
2261                    emisTotSOXLDDV2(j, i) = emisExtrSOXLDDV2(j, i) + emisTransSOXLDDV2(j, i) + emisRefSOXLDDV2(j, i) + emisTSDSOXLDDV2(j, i) + emisOpSO2LDDV2(j, i)
2262                    emisTotSOXHDDV2b(j, i) = emisExtrSOXHDDV2b(j, i) + emisTransSOXHDDV2b(j, i) + emisRefSOXHDDV2b(j, i) + emisTSDSOXHDDV2b(j, i) + emisOpSO2HDDV2b(j, i)
2263
```

```
2264                    'TOTALS:
2265                    gasEmisTotSOXTotal(j, i) = emisTotSOXLDGV(j, i) + emisTotSOXLDGT1(j, i)
    + emisTotSOXLDGT2(j, i) + emisTotSOXHDGV2b(j, i)
2266                    diesEmisTotSOXTotal(j, i) = emisTotSOXLDDV(j, i) + emisTotSOXLDDV1(j, i)
     + emisTotSOXLDDV2(j, i) + emisTotSOXHDDV2b(j, i)
2267                End If
2268            Next
2269        Next
2270
2271        'Economic Benefit/Cost of Total Sulfur Dioxide (SOX) Emissions
2272        Dim damageEmisTotSOXLDGV(MY, CY) As Double
2273        Dim damageEmisTotSOXLDGT1(MY, CY) As Double
2274        Dim damageEmisTotSOXLDGT2(MY, CY) As Double
2275        Dim damageEmisTotSOXHDGV2b(MY, CY) As Double
2276
2277        Dim damageEmisTotSOXLDDV(MY, CY) As Double
2278        Dim damageEmisTotSOXLDDV1(MY, CY) As Double
2279        Dim damageEmisTotSOXLDDV2(MY, CY) As Double
2280        Dim damageEmisTotSOXHDDV2b(MY, CY) As Double
2281
2282        Dim discDamageEmisTotSOXLDGV(MY, CY) As Double
2283        Dim discDamageEmisTotSOXLDGT1(MY, CY) As Double
2284        Dim discDamageEmisTotSOXLDGT2(MY, CY) As Double
2285        Dim discDamageEmisTotSOXHDGV2b(MY, CY) As Double
2286
2287        Dim discDamageEmisTotSOXLDDV(MY, CY) As Double
2288        Dim discDamageEmisTotSOXLDDV1(MY, CY) As Double
2289        Dim discDamageEmisTotSOXLDDV2(MY, CY) As Double
2290        Dim discDamageEmisTotSOXHDDV2b(MY, CY) As Double
2291
2292        'TOTALS:
2293        Dim gasDamageEmisTotSOXTotal(MY, CY) As Double
2294        Dim diesDamageEmisTotSOXTotal(MY, CY) As Double
2295
2296        Dim discGasDamageEmisTotSOXTotal(MY, CY) As Double
2297        Dim discDiesDamageEmisTotSOXTotal(MY, CY) As Double
2298
2299        For i = 0 To CY
2300            For j = 0 To MY
2301                If i >= j Then
2302                    damageEmisTotSOXLDGV(j, i) = emisTotSOXLDGV(j, i) * CAFE.Parameters.
    damageCostSO2 / 1000
2303                    damageEmisTotSOXLDGT1(j, i) = emisTotSOXLDGT1(j, i) * CAFE.Parameters.
    damageCostSO2 / 1000
2304                    damageEmisTotSOXLDGT2(j, i) = emisTotSOXLDGT2(j, i) * CAFE.Parameters.
    damageCostSO2 / 1000
2305                    damageEmisTotSOXHDGV2b(j, i) = emisTotSOXHDGV2b(j, i) * CAFE.Parameters.
    damageCostSO2 / 1000
2306
2307                    damageEmisTotSOXLDDV(j, i) = emisTotSOXLDDV(j, i) * CAFE.Parameters.
    damageCostSO2 / 1000
2308                    damageEmisTotSOXLDDV1(j, i) = emisTotSOXLDDV1(j, i) * CAFE.Parameters.
    damageCostSO2 / 1000
2309                    damageEmisTotSOXLDDV2(j, i) = emisTotSOXLDDV2(j, i) * CAFE.Parameters.
    damageCostSO2 / 1000
2310                    damageEmisTotSOXHDDV2b(j, i) = emisTotSOXHDDV2b(j, i) * CAFE.Parameters.
    damageCostSO2 / 1000
2311
2312                    discDamageEmisTotSOXLDGV(j, i) = damageEmisTotSOXLDGV(j, i) * ((1 + CAFE
    .Parameters.discountRate) ^ -i)
2313                    discDamageEmisTotSOXLDGT1(j, i) = damageEmisTotSOXLDGT1(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
2314                    discDamageEmisTotSOXLDGT2(j, i) = damageEmisTotSOXLDGT2(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
2315                    discDamageEmisTotSOXHDGV2b(j, i) = damageEmisTotSOXHDGV2b(j, i) * ((1 +
    CAFE.Parameters.discountRate) ^ -i)
2316
2317                    discDamageEmisTotSOXLDDV(j, i) = damageEmisTotSOXLDDV(j, i) * ((1 + CAFE
    .Parameters.discountRate) ^ -i)
```

```
                        discDamageEmisTotSOXLDDV1(j, i) = damageEmisTotSOXLDDV1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotSOXLDDV2(j, i) = damageEmisTotSOXLDDV2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotSOXHDDV2b(j, i) = damageEmisTotSOXHDDV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                        'TOTALS:
                        gasDamageEmisTotSOXTotal(j, i) = damageEmisTotSOXLDGV(j, i) + damageEmisTotSOXLDGT1(j, i) + damageEmisTotSOXLDGT2(j, i) + damageEmisTotSOXHDGV2b(j, i)
                        diesDamageEmisTotSOXTotal(j, i) = damageEmisTotSOXLDDV(j, i) + damageEmisTotSOXLDDV1(j, i) + damageEmisTotSOXLDDV2(j, i) + damageEmisTotSOXHDDV2b(j, i)

                        discGasDamageEmisTotSOXTotal(j, i) = discDamageEmisTotSOXLDGV(j, i) + discDamageEmisTotSOXLDGT1(j, i) + discDamageEmisTotSOXLDGT2(j, i) + discDamageEmisTotSOXHDGV2b(j, i)
                        discDiesDamageEmisTotSOXTotal(j, i) = discDamageEmisTotSOXLDDV(j, i) + discDamageEmisTotSOXLDDV1(j, i) + discDamageEmisTotSOXLDDV2(j, i) + discDamageEmisTotSOXHDDV2b(j, i)

                    End If
                Next
            Next


            ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
            ' safety/weight fatalities calculations
            '
            Dim n As Integer

            Dim nSafetyClasses As Integer = 17
            Dim nSWClasses As Integer = 34
            Dim newSafetyFatals(nSafetyClasses)(,) As Double
            Dim newSafetyFatalsTotals(MY, CY) As Double
            'Dim newSwFatals(,) As Double
            Dim swFatals(MY, CY) As Double
            Dim swDeltaCW(MY) As Double

            Dim val As Double

            If CAFE.SafetyIncluded Then
                For n = 1 To nSWClasses
                    k = Math.Ceiling(n / 2)
                    If n Mod 2 = 1 Then
                        ReDim newSafetyFatals(k)(MY, CY)
                    End If
                    'ReDim Preserve newSwFatals(n)(MY, CY)
                    Me.LoadSafetyWeightAggregateArrays(Scenario, MY, CY, swFatals, swDeltaCW, n)

                    For i = 0 To CY
                        For j = 0 To MY
                            If i >= j Then
                                ' calculate the fatals
                                val = swFatals(j, i) * (1 - (GetWPE(n) / 100) * (swDeltaCW(j) / 100)) * 1000

                                ' add fatals for individual classes
                                newSafetyFatals(k)(j, i) += val
                                ' and add them to total fatals--e.g. calculate totals
                                newSafetyFatalsTotals(j, i) += val
                            End If
                        Next j
                    Next i
                Next n
            End If


            '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
        '''''''''''''''''''''
2376        ' Write data to the stack for effects outputs
2377        '
2378        Dim effectStack As EffectStackClass
2379        Dim ModelYear As Integer
2380
2381        If IsNothing(CAFE.EffectsStack(Scenario)) Then
2382            ' EffectsStack for this scenario has not been initialized yet, initialize it
2383            CAFE.EffectsStack(Scenario) = New EffectsStackClass
2384        End If
2385
2386        ' scan each available MY
2387        For i = 0 To MY
2388            If CAFE.ModelYearsSelected(i) Then
2389                ModelYear = CAFE.GetMYearFromIndex(i)
2390
2391                ' copy necessary variables to the stack
2392                For j = 0 To 2
2393                    ' get the data to push onto the stack
2394                    effectStack = New EffectStackClass(Scenario, ModelYear, j)
2395
2396                    If j = 0 Then
2397                        For k = 0 To CY
2398                            ' Energy Consumption
2399                            effectStack.EC_LifetimeFuelConsumption(0) += gasGalTotal(i, k)
2400                            effectStack.EC_LifetimeFuelConsumption(1) += diesGalTotal(i, k)
2401                            effectStack.EC_LifetimeFuelConsumption(2) += (gasGalTotal(i, k)
        + diesGalTotal(i, k))
2402
2403                            ' Emissions
2404                            effectStack.Em_CO2(0) += emisTotCO2LDDV(i, k)
2405                            effectStack.Em_CO2(1) += emisTotCO2LDGV(i, k)
2406                            effectStack.Em_CO2(2) += emisTotCO2LDDV1(i, k)
2407                            effectStack.Em_CO2(3) += emisTotCO2LDGT1(i, k)
2408                            effectStack.Em_CO2(4) += emisTotCO2LDDV2(i, k)
2409                            effectStack.Em_CO2(5) += emisTotCO2LDGT2(i, k)
2410                            effectStack.Em_CO2(6) += emisTotCO2HDDV2b(i, k)
2411                            effectStack.Em_CO2(7) += emisTotCO2HDGV2b(i, k)
2412                            effectStack.Em_CO2(8) += (gasEmisTotCO2Total(i, k) +
        diesEmisTotCO2Total(i, k))
2413
2414                            effectStack.Em_CO(0) += emisTotCOLDDV(i, k)
2415                            effectStack.Em_CO(1) += emisTotCOLDGV(i, k)
2416                            effectStack.Em_CO(2) += emisTotCOLDDV1(i, k)
2417                            effectStack.Em_CO(3) += emisTotCOLDGT1(i, k)
2418                            effectStack.Em_CO(4) += emisTotCOLDDV2(i, k)
2419                            effectStack.Em_CO(5) += emisTotCOLDGT2(i, k)
2420                            effectStack.Em_CO(6) += emisTotCOHDDV2b(i, k)
2421                            effectStack.Em_CO(7) += emisTotCOHDGV2b(i, k)
2422                            effectStack.Em_CO(8) += (gasEmisTotCOTotal(i, k) +
        diesEmisTotCOTotal(i, k))
2423
2424                            effectStack.Em_VOC(0) += emisTotVOCLDDV(i, k)
2425                            effectStack.Em_VOC(1) += emisTotVOCLDGV(i, k)
2426                            effectStack.Em_VOC(2) += emisTotVOCLDDV1(i, k)
2427                            effectStack.Em_VOC(3) += emisTotVOCLDGT1(i, k)
2428                            effectStack.Em_VOC(4) += emisTotVOCLDDV2(i, k)
2429                            effectStack.Em_VOC(5) += emisTotVOCLDGT2(i, k)
2430                            effectStack.Em_VOC(6) += emisTotVOCHDDV2b(i, k)
2431                            effectStack.Em_VOC(7) += emisTotVOCHDGV2b(i, k)
2432                            effectStack.Em_VOC(8) += (gasEmisTotVOCTotal(i, k) +
        diesEmisTotVOCTotal(i, k))
2433
2434                            effectStack.Em_NOX(0) += emisTotNOXLDDV(i, k)
2435                            effectStack.Em_NOX(1) += emisTotNOXLDGV(i, k)
2436                            effectStack.Em_NOX(2) += emisTotNOXLDDV1(i, k)
2437                            effectStack.Em_NOX(3) += emisTotNOXLDGT1(i, k)
2438                            effectStack.Em_NOX(4) += emisTotNOXLDDV2(i, k)
2439                            effectStack.Em_NOX(5) += emisTotNOXLDGT2(i, k)
```

```
2440                            effectStack.Em_NOX(6) += emisTotNOXHDDV2b(i, k)
2441                            effectStack.Em_NOX(7) += emisTotNOXHDGV2b(i, k)
2442                            effectStack.Em_NOX(8) += (gasEmisTotNOXTotal(i, k) +
      diesEmisTotNOXTotal(i, k))
2443
2444                            effectStack.Em_PM(0) += emisTotPMLDDV(i, k)
2445                            effectStack.Em_PM(1) += emisTotPMLDGV(i, k)
2446                            effectStack.Em_PM(2) += emisTotPMLDDV1(i, k)
2447                            effectStack.Em_PM(3) += emisTotPMLDGT1(i, k)
2448                            effectStack.Em_PM(4) += emisTotPMLDDV2(i, k)
2449                            effectStack.Em_PM(5) += emisTotPMLDGT2(i, k)
2450                            effectStack.Em_PM(6) += emisTotPMHDDV2b(i, k)
2451                            effectStack.Em_PM(7) += emisTotPMHDGV2b(i, k)
2452                            effectStack.Em_PM(8) += (gasEmisTotPMTotal(i, k) +
      diesEmisTotPMTotal(i, k))
2453
2454                            effectStack.Em_SOX(0) += emisTotSOXLDDV(i, k)
2455                            effectStack.Em_SOX(1) += emisTotSOXLDGV(i, k)
2456                            effectStack.Em_SOX(2) += emisTotSOXLDDV1(i, k)
2457                            effectStack.Em_SOX(3) += emisTotSOXLDGT1(i, k)
2458                            effectStack.Em_SOX(4) += emisTotSOXLDDV2(i, k)
2459                            effectStack.Em_SOX(5) += emisTotSOXLDGT2(i, k)
2460                            effectStack.Em_SOX(6) += emisTotSOXHDDV2b(i, k)
2461                            effectStack.Em_SOX(7) += emisTotSOXHDGV2b(i, k)
2462                            effectStack.Em_SOX(8) += (gasEmisTotSOXTotal(i, k) +
      diesEmisTotSOXTotal(i, k))
2463
2464                            ' Safety
2465                            If CAFE.SafetyIncluded Then
2466                                ' PREVIOUS CALCS USING SAFETY CLASS AGGREGATION
2467                                '
2468                                'effectStack.Sft_TotalLifetimeFatalities(0) +=
      vSmallCarFatal(i, k)
2469                                'effectStack.Sft_TotalLifetimeFatalities(1) += smallCarFatal
      (i, k)
2470                                'effectStack.Sft_TotalLifetimeFatalities(2) += midCarFatal(i
      , k)
2471                                'effectStack.Sft_TotalLifetimeFatalities(3) += largeCarFatal
      (i, k)
2472                                'effectStack.Sft_TotalLifetimeFatalities(4) +=
      compactPickupFatal(i, k)
2473                                'effectStack.Sft_TotalLifetimeFatalities(5) +=
      largePickupFatal(i, k)
2474                                'effectStack.Sft_TotalLifetimeFatalities(6) +=
      heavyPickupFatal(i, k)
2475                                'effectStack.Sft_TotalLifetimeFatalities(7) +=
      smallUtilFatal(i, k)
2476                                'effectStack.Sft_TotalLifetimeFatalities(8) += midUtilFatal
      (i, k)
2477                                'effectStack.Sft_TotalLifetimeFatalities(9) +=
      largeUtilFatal(i, k)
2478                                'effectStack.Sft_TotalLifetimeFatalities(10) += minivanFatal
      (i, k)
2479                                'effectStack.Sft_TotalLifetimeFatalities(11) +=
      largeVanFatal(i, k)
2480                                'effectStack.Sft_TotalLifetimeFatalities(12) +=
      sportyCarFatal(i, k)
2481                                'effectStack.Sft_TotalLifetimeFatalities(13) +=
      muscleCarFatal(i, k)
2482                                'effectStack.Sft_TotalLifetimeFatalities(14) +=
      economyCarFatal(i, k)
2483                                'effectStack.Sft_TotalLifetimeFatalities(15) +=
      otherCarFatal(i, k)
2484                                'effectStack.Sft_TotalLifetimeFatalities(16) +=
      twoDoorUtilFatal(i, k)
2485                                'effectStack.Sft_TotalLifetimeFatalities(17) += (mixFatal(i,
       k) + weightFatal(i, k))
2486
2487                                ' NEW CALCS USING SAFETY/WEIGHT CLASS AGGREGATION
```

```
2488                                '
2489                                effectStack.Sft_TotalLifetimeFatalities(0) += newSafetyFatals(1)(i, k)
2490                                effectStack.Sft_TotalLifetimeFatalities(1) += newSafetyFatals(2)(i, k)
2491                                effectStack.Sft_TotalLifetimeFatalities(2) += newSafetyFatals(3)(i, k)
2492                                effectStack.Sft_TotalLifetimeFatalities(3) += newSafetyFatals(4)(i, k)
2493                                effectStack.Sft_TotalLifetimeFatalities(4) += newSafetyFatals(5)(i, k)
2494                                effectStack.Sft_TotalLifetimeFatalities(5) += newSafetyFatals(6)(i, k)
2495                                effectStack.Sft_TotalLifetimeFatalities(6) += newSafetyFatals(7)(i, k)
2496                                effectStack.Sft_TotalLifetimeFatalities(7) += newSafetyFatals(8)(i, k)
2497                                effectStack.Sft_TotalLifetimeFatalities(8) += newSafetyFatals(9)(i, k)
2498                                effectStack.Sft_TotalLifetimeFatalities(9) += newSafetyFatals(10)(i, k)
2499                                effectStack.Sft_TotalLifetimeFatalities(10) += newSafetyFatals(11)(i, k)
2500                                effectStack.Sft_TotalLifetimeFatalities(11) += newSafetyFatals(12)(i, k)
2501                                effectStack.Sft_TotalLifetimeFatalities(12) += newSafetyFatals(13)(i, k)
2502                                effectStack.Sft_TotalLifetimeFatalities(13) += newSafetyFatals(14)(i, k)
2503                                effectStack.Sft_TotalLifetimeFatalities(14) += newSafetyFatals(15)(i, k)
2504                                effectStack.Sft_TotalLifetimeFatalities(15) += newSafetyFatals(16)(i, k)
2505                                effectStack.Sft_TotalLifetimeFatalities(16) += newSafetyFatals(17)(i, k)
2506
2507                                effectStack.Sft_TotalLifetimeFatalities(17) += newSafetyFatalsTotals(i, k)
2508                            End If
2509
2510                            ' Undiscounted Owner and Societal Costs
2511                            '
2512                            'effectStack.TotalLifetimePretaxFuelExpenditures += (gasGalExpTotal(i, k) + diesGalExpTotal(i, k))
2513                            'effectStack.FuelTaxRevenues += (gasTaxTotal(i, k) + diesTaxTotal(i, k))
2514                            'effectStack.TravelValue += (gasDriveSurplusTotal(i, k) + diesDriveSurplusTotal(i, k))
2515                            'effectStack.RefuelingTimeValue += (gasRefuelSurplusTotal(i, k) + diesRefuelSurplusTotal(i, k))
2516                            ' /\
2517                            '  ||  - the above stuff was moved to veh and effects stack
2518                            '
2519                            effectStack.PetroleumMarketExternalities += (gasTotalEconCostTotal(i, k) + diesTotalEconCostTotal(i, k))
2520                            effectStack.CongestionCosts += (gasCongestionTotal(i, k) + diesCongestionTotal(i, k))
2521                            effectStack.NoiseCosts += (gasNoiseTotal(i, k) + diesNoiseTotal(i, k))
2522                            effectStack.AccidentCosts += (gasAccidentTotal(i, k) + diesAccidentTotal(i, k))
2523                            effectStack.CO2 += (gasDamageEmisTotCO2Total(i, k) + diesDamageEmisTotCO2Total(i, k))
2524                            effectStack.CO += (gasDamageEmisTotCOTotal(i, k) + diesDamageEmisTotCOTotal(i, k))
2525                            effectStack.VOC += (gasDamageEmisTotVOCTotal(i, k) + diesDamageEmisTotVOCTotal(i, k))
2526                            effectStack.NOX += (gasDamageEmisTotNOXTotal(i, k) + diesDamageEmisTotNOXTotal(i, k))
```

```
2527                            effectStack.PM += (gasDamageEmisTotPMTotal(i, k) +
    diesDamageEmisTotPMTotal(i, k))
2528                            effectStack.SOX += (gasDamageEmisTotSOXTotal(i, k) +
    diesDamageEmisTotSOXTotal(i, k))
2529                            effectStack.MonetizedFatalities = Double.Epsilon
2530
2531                            ' Discounted Owner and Societal Costs
2532                            '
2533                            'effectStack.DiscTotalLifetimePretaxFuelExpenditures +=
    (discGasGalExpTotal(i, k) + discDiesGalExpTotal(i, k))
2534                            'effectStack.DiscFuelTaxRevenues += (discGasTaxTotal(i, k) +
    discDiesTaxTotal(i, k))
2535                            'effectStack.DiscTravelValue += (discGasDriveSurplusTotal(i, k)
    + discDiesDriveSurplusTotal(i, k))
2536                            'effectStack.DiscRefuelingTimeValue +=
    (discGasRefuelSurplusTotal(i, k) + discDiesRefuelSurplusTotal(i, k))
2537                            '  /\
2538                            '  ||  - the above stuff was moved to veh and effects stack
2539                            '
2540                            effectStack.DiscPetroleumMarketExternalities +=
    (discGasTotalEconCostTotal(i, k) + discDiesTotalEconCostTotal(i, k))
2541                            effectStack.DiscCongestionCosts += (discGasCongestionTotal(i, k)
     + discDiesCongestionTotal(i, k))
2542                            effectStack.DiscNoiseCosts += (discGasNoiseTotal(i, k) +
    discDiesNoiseTotal(i, k))
2543                            effectStack.DiscAccidentCosts += (discGasAccidentTotal(i, k) +
    discDiesAccidentTotal(i, k))
2544                            effectStack.DiscCO2 += (discGasDamageEmisTotCO2Total(i, k) +
    discDiesDamageEmisTotCO2Total(i, k))
2545                            effectStack.DiscCO += (discGasDamageEmisTotCOTotal(i, k) +
    discDiesDamageEmisTotCOTotal(i, k))
2546                            effectStack.DiscVOC += (discGasDamageEmisTotVOCTotal(i, k) +
    discDiesDamageEmisTotVOCTotal(i, k))
2547                            effectStack.DiscNOX += (discGasDamageEmisTotNOXTotal(i, k) +
    discDiesDamageEmisTotNOXTotal(i, k))
2548                            effectStack.DiscPM += (discGasDamageEmisTotPMTotal(i, k) +
    discDiesDamageEmisTotPMTotal(i, k))
2549                            effectStack.DiscSOX += (discGasDamageEmisTotSOXTotal(i, k) +
    discDiesDamageEmisTotSOXTotal(i, k))
2550                            effectStack.DiscMonetizedFatalities = Double.Epsilon
2551                        Next k
2552                        ' Report data to the "Summary"
2553                        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
    SummaryTypeEnum.TotalLifeTimeFatalities, _
2554                                                         Scenario, CAFE.GetMYearFromIndex(i),
     -1, effectStack.Sft_TotalLifetimeFatalities(17))
2555                        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
    SummaryTypeEnum.LifeTimeFuelConsumption, _
2556                                                         Scenario, CAFE.GetMYearFromIndex(i),
     -1, effectStack.EC_LifetimeFuelConsumption(2))
2557                        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
    SummaryTypeEnum.UndiscTotalFuelExpend, _
2558                                                         Scenario, CAFE.GetMYearFromIndex(i),
     -1, effectStack.TotalLifetimePretaxFuelExpenditures)
2559                        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
    SummaryTypeEnum.UndiscFuelTaxRevenues, _
2560                                                         Scenario, CAFE.GetMYearFromIndex(i),
     -1, effectStack.FuelTaxRevenues)
2561                        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
    SummaryTypeEnum.UndiscTravelValue, _
2562                                                         Scenario, CAFE.GetMYearFromIndex(i),
     -1, effectStack.TravelValue)
2563                        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
    SummaryTypeEnum.UndiscRefuelingTimeValue, _
2564                                                         Scenario, CAFE.GetMYearFromIndex(i),
     -1, effectStack.RefuelingTimeValue)
2565                        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
    SummaryTypeEnum.UndiscPetroMarketExternalities, _
2566                                                         Scenario, CAFE.GetMYearFromIndex(i),
```

```
-1, effectStack.PetroleumMarketExternalities)
2567                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.UndiscCongestionCosts, _
2568                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.CongestionCosts)
2569                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.UndiscNoiseCosts, _
2570                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.NoiseCosts)
2571                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.UndiscAccidentCosts, _
2572                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.AccidentCosts)
2573                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.UndiscCO2, _
2574                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.CO2)
2575                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.UndiscCO, _
2576                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.CO)
2577                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.UndiscVOC, _
2578                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.VOC)
2579                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.UndiscNOX, _
2580                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.NOX)
2581                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.UndiscPM, _
2582                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.PM)
2583                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.UndiscSOX, _
2584                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.SOX)
2585
2586                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscTotalFuelExpend, _
2587                                Scenario, CAFE.GetMYearFromIndex(i), -1, effectStack.
DiscTotalLifetimePretaxFuelExpenditures)
2588                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscFuelTaxRevenues, _
2589                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.DiscFuelTaxRevenues)
2590                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscTravelValue, _
2591                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.DiscTravelValue)
2592                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscRefuelingTimeValue, _
2593                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.DiscRefuelingTimeValue)
2594                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscPetroMarketExternalities, _
2595                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.DiscPetroleumMarketExternalities)
2596                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscCongestionCosts, _
2597                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.DiscCongestionCosts)
2598                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscNoiseCosts, _
2599                                                     Scenario, CAFE.GetMYearFromIndex(i),
-1, effectStack.DiscNoiseCosts)
2600                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscAccidentCosts, _
2601                                                     Scenario, CAFE.GetMYearFromIndex(i),
```

```
 -1, effectStack.DiscAccidentCosts)
                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscCO2, _
                                                     Scenario, CAFE.GetMYearFromIndex(i),
 -1, effectStack.DiscCO2)
                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscCO, _
                                                     Scenario, CAFE.GetMYearFromIndex(i),
 -1, effectStack.DiscCO)
                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscVOC, _
                                                     Scenario, CAFE.GetMYearFromIndex(i),
 -1, effectStack.DiscVOC)
                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscNOX, _
                                                     Scenario, CAFE.GetMYearFromIndex(i),
 -1, effectStack.DiscNOX)
                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscPM, _
                                                     Scenario, CAFE.GetMYearFromIndex(i),
 -1, effectStack.DiscPM)
                    CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.
SummaryTypeEnum.DiscSOX, _
                                                     Scenario, CAFE.GetMYearFromIndex(i),
 -1, effectStack.DiscSOX)

                End If

                ' convert all NaN, -Inf, and +Inf values to 0.0 and convert Double.
Epsilon values to NaN
                effectStack.RemoveNaNs()

                ' push the data onto the stack
                If CAFE.OperatingModes.MonteCarloAnalysis = False Then
                    ' Don't push the stack if in MonteCarlo mode
                    CAFE.EffectsStack(Scenario).Push(effectStack)
                Else
                    effectStack = Nothing       ' Monte Carlo mode -- clear out the
stack element
                End If
            Next j
        End If
    Next i

    '
    ' Write data to the stack for environmental assessment outputs
    '
    Dim eaStack As EAStackClass
    Dim ea_nMY As Integer
    Dim ea_nCY As Integer

    Dim ea_firstMY, ea_lastMY As Integer

    ' find first MY
    For i = 0 To MY
        If CAFE.ModelYearsSelected(i) Then
            ea_firstMY = i
            Exit For
        End If
    Next i

    ' find last MY
    For i = MY To 0 Step -1
        If CAFE.ModelYearsSelected(i) Then
            ea_lastMY = i
            Exit For
        End If
    Next i
```

```
2656
2657        ea_nMY = ea_lastMY - ea_firstMY + 1
2658        ea_nCY = ea_nMY + 25
2659
2660        If IsNothing(CAFE.EAsStack(Scenario)) Then
2661            ' EAsStack for this scenario has not been initialized yet, initialize it
2662            CAFE.EAsStack(Scenario) = New EAsStackClass
2663        End If
2664
2665        eaStack = New EAStackClass(ea_nMY + 1, ea_nCY + 1)      ' +1 for totals
2666
2667        For i = ea_firstMY To ea_lastMY
2668            For j = ea_firstMY To ea_lastMY + 25
2669                eaStack.FC(i - ea_firstMY)(j - ea_firstMY) = gasGalTotal(i, j) + diesGalTotal(i, j)
2670                eaStack.FC(i - ea_firstMY)(ea_nCY) += gasGalTotal(i, j) + diesGalTotal(i, j)
2671                eaStack.FC(ea_nMY)(j - ea_firstMY) += gasGalTotal(i, j) + diesGalTotal(i, j)
2672
2673                eaStack.CO(i - ea_firstMY)(j - ea_firstMY) = gasEmisTotCOTotal(i, j) + diesEmisTotCOTotal(i, j)
2674                eaStack.CO(i - ea_firstMY)(ea_nCY) += gasEmisTotCOTotal(i, j) + diesEmisTotCOTotal(i, j)
2675                eaStack.CO(ea_nMY)(j - ea_firstMY) += gasEmisTotCOTotal(i, j) + diesEmisTotCOTotal(i, j)
2676
2677                eaStack.VOC(i - ea_firstMY)(j - ea_firstMY) = gasEmisTotVOCTotal(i, j) + diesEmisTotVOCTotal(i, j)
2678                eaStack.VOC(i - ea_firstMY)(ea_nCY) += gasEmisTotVOCTotal(i, j) + diesEmisTotVOCTotal(i, j)
2679                eaStack.VOC(ea_nMY)(j - ea_firstMY) += gasEmisTotVOCTotal(i, j) + diesEmisTotVOCTotal(i, j)
2680
2681                eaStack.NOX(i - ea_firstMY)(j - ea_firstMY) = gasEmisTotNOXTotal(i, j) + diesEmisTotNOXTotal(i, j)
2682                eaStack.NOX(i - ea_firstMY)(ea_nCY) += gasEmisTotNOXTotal(i, j) + diesEmisTotNOXTotal(i, j)
2683                eaStack.NOX(ea_nMY)(j - ea_firstMY) += gasEmisTotNOXTotal(i, j) + diesEmisTotNOXTotal(i, j)
2684
2685                eaStack.SO2(i - ea_firstMY)(j - ea_firstMY) = gasEmisTotSOXTotal(i, j) + diesEmisTotSOXTotal(i, j)
2686                eaStack.SO2(i - ea_firstMY)(ea_nCY) += gasEmisTotSOXTotal(i, j) + diesEmisTotSOXTotal(i, j)
2687                eaStack.SO2(ea_nMY)(j - ea_firstMY) += gasEmisTotSOXTotal(i, j) + diesEmisTotSOXTotal(i, j)
2688
2689                eaStack.PM(i - ea_firstMY)(j - ea_firstMY) = gasEmisTotPMTotal(i, j) + diesEmisTotPMTotal(i, j)
2690                eaStack.PM(i - ea_firstMY)(ea_nCY) += gasEmisTotPMTotal(i, j) + diesEmisTotPMTotal(i, j)
2691                eaStack.PM(ea_nMY)(j - ea_firstMY) += gasEmisTotPMTotal(i, j) + diesEmisTotPMTotal(i, j)
2692
2693                eaStack.CO2(i - ea_firstMY)(j - ea_firstMY) = gasEmisTotCO2Total(i, j) + diesEmisTotCO2Total(i, j)
2694                eaStack.CO2(i - ea_firstMY)(ea_nCY) += gasEmisTotCO2Total(i, j) + diesEmisTotCO2Total(i, j)
2695                eaStack.CO2(ea_nMY)(j - ea_firstMY) += gasEmisTotCO2Total(i, j) + diesEmisTotCO2Total(i, j)
2696            Next j
2697        Next i
2698
2699        CAFE.EAsStack(Scenario).Push(eaStack)
2700    End Function
2701
2702    Private Sub LoadMobile6AggregateArrays(ByVal Scenario As Integer, ByVal MY As Integer, ByVal CY As Integer, _
2703                                           ByRef actBaseStd(,) As Double, ByRef actAltStd(,) As Double, _
```

```
2704                                        ByRef baseCPM(,) As Double, ByRef altCPM(,) As
    Double, _
2705                                        ByRef VMT(,) As Double, ByRef Gallons(,) As
    Double, _
2706                                        ByVal M6ClassIdx As Integer)
2707        '
2708        Dim i, j As Integer
2709
2710        ' load data for actAltStd, actBaseStd, altCPM, baseCPM, and VMT using the aggregate
    data
2711        Dim M6ClassesBase() As AggregateVehiclesClass.MyData
2712        Dim M6ClassesAlt() As AggregateVehiclesClass.MyData
2713
2714        ' scan each MY
2715        For i = 0 To MY
2716            If Not IsNothing(CAFE.BaselineAggregateData(i)) Then
2717                ' get the base values
2718                M6ClassesBase = CAFE.BaselineAggregateData(i).GetMobile6Classes()
2719                M6ClassesAlt = CAFE.ScenarioAggregateData(i).GetMobile6Classes()
2720
2721                ' scan each CY (years 0 to 25)
2722                For j = 0 To 25
2723                    ' load base values
2724                    actBaseStd(i, i + j) = M6ClassesBase(M6ClassIdx).FuelEconomy
2725                    baseCPM(i, i + j) = M6ClassesBase(M6ClassIdx).CPM(j)
2726                    ' load alt values
2727                    actAltStd(i, i + j) = M6ClassesAlt(M6ClassIdx).FuelEconomy
2728                    altCPM(i, i + j) = M6ClassesAlt(M6ClassIdx).CPM(j)
2729                    VMT(i, i + j) = M6ClassesAlt(M6ClassIdx).VMT(j)
2730                    Gallons(i, i + j) = M6ClassesAlt(M6ClassIdx).Gallons(j)
2731                Next j
2732            End If
2733        Next i
2734    End Sub
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
```

```
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815 End Class
2816
```

```
1  Imports Microsoft.Win32
2  Imports System.IO
3
4  Public Class CafeForm
5      Inherits System.Windows.Forms.Form
6
7   Windows Form Designer generated code
8
9      Public MyCafeClass As CAFE
10     Private QuickExit As Boolean
11     Public VersionString As String
12     Public VersionMajor As Integer
13     Public VersionMinor As Integer
14     Public VersionDate As String = "04/27/05"
15
16     'Private ModelRunning As Boolean        ' ModelRunning was moved to CAFE class
17
18     ' Classes which handle input files.
19     Public RawMarketData As RawMarketDataClass
20     Public RawVehicles As RawVehiclesClass
21     Public RawEngines As RawEnginesClass
22     Public RawTransmissions As RawTransmissionsClass
23     Public RawTechnology As RawTechnologies
24     Public RawParameters As RawParametersClass
25     Public RawScenarios As RawScenariosClass
26
27     ' File names brought in from the registry
28     Private RegMarketDataFile As String
29     Private RegVehFile As String
30     Private RegEngFile As String
31     Private RegTranFile As String
32     Private RegTechFile As String
33     Private RegParamFile As String
34     Private RegScenFile As String
35
36     ' Options brought in from the registry
37     Private RegEnteredOK As Boolean     ' Indicates we were able to read settings from
       registry
38     Private RegNoFines As Boolean
39     Private RegLowCostFirst As Integer
40     Private RegAutoToggle As Boolean
41     Private RegBackFill As Boolean
42     Private RegStageAnalysis As Boolean
43     Private RegMonteCarloAnalysis As Boolean
44     Private RegMonteCarloIterations As Integer
45     Private RegNewEngAuto As Boolean    ' "Create" a new engine on Auto/Truck mix
46     Private RegNewEngRC As Boolean      ' Create a new engine on multi-RC mix
47     Private RegNewEngEnable As Boolean  ' Create a new engone on mixed tech-enable states
48     Private RegNewTranAuto As Boolean   ' "Create" a new transmission on Auto/Truck mix
49     Private RegNewTranRC As Boolean     ' Create a new transmission on multi-RC mix
50     Private RegNewTranEnable As Boolean ' Create a new transmission on mixed tech-enable
       states
51     Private RegSkipAuto As Boolean
52     Private RegSkipRC As Boolean
53     Private RegSkipEnable As Boolean
54     Private RegReserveTechs As Boolean  ' Reserve technologies if in compliance
55
56     Private RegPath As Integer
57     Private RegCost As Integer  ' Tech Cost Estimates: 1 = low; 2 = med; 3 = high
58     Private RegFC As Integer    ' Tech FuelConsumption Estimates: 1 = low; 2 = med; 3 =
       high
59
60     ' Report options from registry
61     Private RegOutputHideUnusedRcs As Boolean
62     Private RegOutputInd As Boolean
63     Private RegOutputMfgs As Boolean
64     Private RegOutputVehs As Boolean
65     Private RegOutputEffs As Boolean
66     Private RegOutputEAs As Boolean
67
```

```
68      ' Input file names (full path name)
69      Public NHTSA_MarketDataFileName As String
70      Public NHTSA_VehiclesFileName As String
71      Public NHTSA_EnginesFileName As String
72      Public NHTSA_TransmissionsFileName As String
73      Public TechnologyFileName As String
74      Public ParametersFileName As String
75      Public ScenariosFileName As String
76
77      Public ModelStartTime As DateTime
78
79      ' Event handlers
80      Private Sub CafeForm_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
        MyBase.Load
81          Dim WbC As Integer
82          Dim TmpStr As String
83          Dim i As Integer
84          Dim j As Integer
85
86          QuickExit = True
87          CAFE.ModelRunning = False
88          CAFE.PathSettings = CAFE.PathsEnum.Path2
89          'CAFE.ModelYearsAvailable = New ArrayList
90          CAFE.FuelConsumptionSavingsEstimates = CAFE.LowAverageHigh.Average
91          CAFE.TechnologyCostEstimates = CAFE.LowAverageHigh.Average
92
93          m_MenuOptionsMY.Enabled = False      ' disable the "Model Year Settings"
94          m_ContextOptionsMY.Enabled = False   ' menu options
95
96          AddHandler Application.Idle, AddressOf OnApplicationIdle     ' Idle time...
97
98          Trace.WriteLine("")
99          Trace.WriteLine("")
100         Trace.WriteLine("Starting new run: " & Now())
101
102         VersionMajor = System.Diagnostics.FileVersionInfo.GetVersionInfo(System.Reflection.
        Assembly.GetExecutingAssembly().Location).FileMajorPart
103         VersionMinor = System.Diagnostics.FileVersionInfo.GetVersionInfo(System.Reflection.
        Assembly.GetExecutingAssembly().Location).FileMinorPart
104
105         VersionString = VersionMajor.ToString + "." + VersionMinor.ToString + " " +
        VersionDate
106         SetStatusBarRevision(VersionString)
107
108         MyCafeClass = New CAFE
109         CAFE.WeStartedExcel = False
110         Dim TempExcelApp As Excel.Application
111         Dim ExcelAlreadyRunning As Boolean = True
112         Try
113             TempExcelApp = GetObject(, "Excel.Application")     'look too see if Excel is
        already running
114         Catch eex As Exception
115             ExcelAlreadyRunning = False
116         End Try
117         If ExcelAlreadyRunning Then
118             WbC = TempExcelApp.Workbooks.Count
119             Select Case WbC
120                 Case 0
121                     TmpStr = "Excel is running (with no workbooks). This may cause problems
        ."
122                 Case 1
123                     TmpStr = "Excel is running with the following workbook:" _
124                       & Chr(13) & "  - " & TempExcelApp.Workbooks(1).Name _
125                       & Chr(13) & "This may cause problems."
126                 Case Else
127                     TmpStr = "Excel is running with the following workbooks:"
128                     j = WbC
129                     If j > 8 Then j = 8
130                     For i = 1 To j
131                         TmpStr = TmpStr & Chr(13) & "  - " & TempExcelApp.Workbooks(i).Name
```

```
132                     Next i
133                     If WbC > j Then TmpStr = TmpStr & Chr(13) & "  - ..."
134                     TmpStr = TmpStr & Chr(13) & "This may cause problems."
135             End Select
136
137             ' If excel was already running, warn the user
138             Me.m_WarningProvider.SetError(Me.m_LabelDummy, TmpStr)
139
140             TempExcelApp = Nothing
141         End If
142
143         ' Now start up our own version of Excel...
144         ExcelApp = CreateObject("Excel.Application")      'run it
145         CAFE.WeStartedExcel = True
146
147         ' Check if there's any filenames lingering on the registry
148         Dim rkey As RegistryKey = Me.OpenRegKey()
149
150         Dim NoReg As Boolean = False     ' flag indicates that registry DOESN'T have our file info
151         Dim NoRegSettings As Boolean = False ' indicates it doesn't have settings info
152         Dim BF As Boolean
153         Dim NF As Boolean
154         Dim Path As CAFE.PathsEnum
155         Dim Cost As CAFE.LowAverageHigh
156
157         Try
158             ' get the default output path
159             CAFE.OutputSettings.DefaultOutputPath = CStr(rkey.OpenSubKey("Output Settings").GetValue("DefaultOutputPath"))
160         Catch ex As Exception
161             CAFE.OutputSettings.DefaultOutputPath = ""
162         End Try
163
164         ' get the templates
165         With CAFE.TemplateSettings
166             Dim TemplatePath As String = Application.StartupPath.Substring(0, Application.StartupPath.LastIndexOf("\")) + "\data"
167
168             Try
169                 .UseTemplates = CBool(rkey.OpenSubKey("Template Settings").GetValue("UseTemplates", True))
170                 .DefaultTemplatePath = CStr(rkey.OpenSubKey("Template Settings").GetValue("DefaultTemplatePath", TemplatePath + "\Template.xls"))
171             Catch ex As Exception
172                 .UseTemplates = True
173                 .DefaultTemplatePath = TemplatePath + "\Template.xls"
174             End Try
175
176             ' Final check ... if registry key exsted but was empty, set default values...
177             If .DefaultTemplatePath = String.Empty Then
178                 .DefaultTemplatePath = TemplatePath + "\Template.xls"
179             End If
180
181             ' check if template files exist and convert relative paths to absolute paths
182             Dim fi As New FileInfo(.DefaultTemplatePath) : .DefaultTemplatePath = fi.FullName()
183
184             ' check if template files exists
185             If Not File.Exists(.DefaultTemplatePath) Then
186                 MsgBox("Missing template file!", MsgBoxStyle.Critical)
187                 .DefaultTemplatePath = ""
188                 .UseTemplates = False
189             End If
190         End With
191
192         With CAFE.OtherSettings
193             Try
194                 ' get whether or not to display the warning message
195                 .NoStartupWarning = CBool(rkey.GetValue("NoStartupWarning"))
```

```
196             Catch ex As Exception
197                 .NoStartupWarning = False
198             End Try
199
200             Try
201                 ' get whether or not to run the effects model
202                 .RunEffectsModel = CBool(rkey.GetValue("RunEffectsModel"))
203             Catch ex As Exception
204                 .RunEffectsModel = False
205             End Try
206
207             Try
208                 ' get whether to use single-file or 3-file dataset input
209                 If IsNothing(rkey.GetValue("SingleFileDatasetInput")) Then
210                     .SingleFileDatasetInput = True
211                 Else
212                     .SingleFileDatasetInput = rkey.GetValue("SingleFileDatasetInput")
213                 End If
214             Catch ex As Exception
215                 .SingleFileDatasetInput = True
216             End Try
217         End With
218
219         Try
220             Me.RegMarketDataFile = CStr(rkey.GetValue("MarketDataDatasetFileName"))
221             Me.RegVehFile = CStr(rkey.GetValue("VehiclesDatasetFileName"))
222             Me.RegEngFile = CStr(rkey.GetValue("EnginesDatasetFileName"))
223             Me.RegTranFile = CStr(rkey.GetValue("TransmissionsDatasetFileName"))
224             Me.RegTechFile = CStr(rkey.GetValue("TechnologyFileName"))
225             Me.RegParamFile = CStr(rkey.GetValue("ParametersFileName"))
226             Me.RegScenFile = CStr(rkey.GetValue("ScenariosFileName"))
227         Catch ex As Exception
228             NoReg = True
229         End Try
230
231         Try
232             ' Now get settings -- record them right onto the form.
233             Me.RegNoFines = CBool(rkey.GetValue("NoFines"))
234             Me.RegLowCostFirst = CInt(rkey.GetValue("LowCostFirst"))
235             Me.RegAutoToggle = CBool(rkey.GetValue("AutoToggleApproach"))
236             Me.RegBackFill = CBool(rkey.GetValue("BackFillTechnology"))
237             Me.RegStageAnalysis = CBool(rkey.GetValue("StageAnalysis"))
238             Me.RegMonteCarloAnalysis = CBool(rkey.GetValue("MonteCarloAnalysis"))
239             Me.RegMonteCarloIterations = CInt(rkey.GetValue("MonteCarloIterations"))
240             Me.RegNewEngAuto = CBool(rkey.GetValue("NewEngineAuto"))
241             Me.RegNewEngRC = CBool(rkey.GetValue("NewEngineRC"))
242             Me.RegNewEngEnable = CBool(rkey.GetValue("NewEngineEnable"))
243             Me.RegNewTranAuto = CBool(rkey.GetValue("NewTranAuto"))
244             Me.RegNewTranRC = CBool(rkey.GetValue("NewTranRC"))
245             Me.RegNewTranEnable = CBool(rkey.GetValue("NewTranEnable"))
246             Me.RegSkipAuto = CBool(rkey.GetValue("SkipAuto"))
247             Me.RegSkipRC = CBool(rkey.GetValue("SkipRC"))
248             Me.RegSkipEnable = CBool(rkey.GetValue("SkipEnable"))
249             Me.RegReserveTechs = CBool(rkey.GetValue("ReserveTechs"))
250
251             Me.RegPath = CInt(rkey.GetValue("NRCPath"))
252             Me.RegCost = CInt(rkey.GetValue("NRCCost"))
253             Me.RegFC = CInt(rkey.GetValue("NRCFuelConsumption"))
254
255             Me.RegEnteredOK = True
256         Catch ex As Exception
257             Me.RegEnteredOK = False
258         End Try
259
260         Try
261             ' get the report settings
262             Me.RegOutputHideUnusedRcs = CBool(rkey.OpenSubKey("Output Settings").GetValue( ↙
    "HideUnusedRCs", True))
263             Me.RegOutputInd = CBool(rkey.OpenSubKey("Output Settings").GetValue( ↙
    "GenIndustryReport"))
```

```
264             Me.RegOutputMfgs = CBool(rkey.OpenSubKey("Output Settings").GetValue(
    "GenMfgsReport"))
265             Me.RegOutputVehs = CBool(rkey.OpenSubKey("Output Settings").GetValue(
    "GenVehiclesReport"))
266             Me.RegOutputEffs = CBool(rkey.OpenSubKey("Output Settings").GetValue(
    "GenEffectsReport"))
267             Me.RegOutputEAs = CBool(rkey.OpenSubKey("Output Settings").GetValue(
    "GenEAReport"))
268         Catch ex As Exception
269             Me.RegOutputHideUnusedRcs = True
270             Me.RegOutputInd = False
271             Me.RegOutputMfgs = False
272             Me.RegOutputVehs = False
273             Me.RegOutputEffs = False
274             Me.RegOutputEAs = False
275         End Try
276
277         ' if NoReg is false (e.g. data found in registry),
278         ' check if registry files exist on disk
279         If Not NoReg Then
280             Me.CheckRegFiles()
281         End If
282
283         ' Save form pointer
284         CAFE.CAFE_Form = Me
285
286         If Not CAFE.OtherSettings.NoStartupWarning Then
287             'Me.Warning() ' Require user to acknowledge warning.
288         End If
289
290         '#############################################################################'
291         '
292         ' set various "hidden" model options to true
293         ' the hidden options should always be true, since they are not displayed to
294         ' the user
295         '
296         '#############################################################################'
297         With CAFE.OtherSettings
298             .RunEffectsModel = True
299             .SingleFileDatasetInput = True
300         End With
301     End Sub
302
303     '''
304     ' Checks if the registry files exist in the directories specified by their paths, and
    if the
305     ' user has access to them.
306     '
307     Private Sub CheckRegFiles()
308         ' dataset files
309         Me.CheckRegFile(Me.RegMarketDataFile)
310         Me.CheckRegFile(Me.RegVehFile)
311         Me.CheckRegFile(Me.RegEngFile)
312         Me.CheckRegFile(Me.RegTranFile)
313
314         ' techs, params, and scenarios files
315         Me.CheckRegFile(Me.RegTechFile)
316         Me.CheckRegFile(Me.RegParamFile)
317         Me.CheckRegFile(Me.RegScenFile)
318     End Sub
319
320     '''
321     ' Checks if the specified file exists and if the user has access to it.  If the file
    does not
322     ' exist, or the user does not have read permissions to it, that file is set to blank.
323     '
324     Private Sub CheckRegFile(ByRef RegFileName As String)
325         If Not File.Exists(RegFileName) Then
326             RegFileName = ""
327         End If
```

```
328     End Sub
329
330     Private Sub CafeForm_Closing(ByVal sender As Object, ByVal e As System.ComponentModel.
        CancelEventArgs) Handles MyBase.Closing
331         If QuickExit = False Then
332             Dim result As MsgBoxResult
333             result = MsgBox("Are you sure you want to exit?", MsgBoxStyle.YesNo, "Confirm
        Exit")
334             If result = MsgBoxResult.No Then
335                 e.Cancel = True
336             End If
337         End If
338         If e.Cancel = False Then
339             If CAFE.WeStartedExcel = True And IsNothing(ExcelApp) = False Then
340                 ExcelApp.Quit()
341             End If
342         End If
343     End Sub
344
345     Private Sub OnApplicationIdle(ByVal sender As Object, ByVal e As EventArgs)
346         Dim b0 As Boolean
347         Dim b1 As Boolean
348         b0 = (Not IsNothing(CAFE.MasterVehicles)) And _
349               (Not IsNothing(CAFE.MasterEngines)) And _
350               (Not IsNothing(CAFE.MasterTransmissions)) And _
351               (Not IsNothing(CAFE.Technologies)) And _
352               (Not IsNothing(CAFE.Parameters)) And _
353               (Not IsNothing(CAFE.Scenarios))
354
355         If b0 Then
356             b1 =
357                 (CAFE.MasterVehicles.Length > 0) And _
358                 (CAFE.MasterEngines.Length > 0) And _
359                 (CAFE.MasterTransmissions.Length > 0) And _
360                 (CAFE.Technologies.nVehicleClasses > 0)
361         Else
362             b1 = False
363         End If
364
365         m_TBarStartModel.Enabled = b0 And b1
366         Me.m_TBarStartModel.Pushed = CAFE.ModelRunning
367         Me.m_TBarCancelModeling.Enabled = CAFE.ModelRunning
368
369         ' show/hide file menu items
370         With CAFE.OtherSettings
371             ' dataset input files
372             '  > file menu
373             Me.FileMenuSeparator00.Visible = False
374             Me.m_MainMarketDataFile.Visible = .SingleFileDatasetInput
375             Me.m_MainModelsFile.Visible = Not .SingleFileDatasetInput
376             Me.m_MainEnginesFile.Visible = Not .SingleFileDatasetInput
377             Me.m_MainTransmissionsFile.Visible = Not .SingleFileDatasetInput
378             '  > context menu
379             Me.m_ContextOpenMarketData.Visible = .SingleFileDatasetInput
380             Me.m_ContextOpenModels.Visible = Not .SingleFileDatasetInput
381             Me.m_ContextOpenEngines.Visible = Not .SingleFileDatasetInput
382             Me.m_ContextOpenTransmissions.Visible = Not .SingleFileDatasetInput
383
384             If .SingleFileDatasetInput Then
385                 Me.m_MainOpenLast.Enabled = (Me.RegMarketDataFile <> "")
386             Else
387                 Me.m_MainOpenLast.Enabled = (Me.RegVehFile <> "" AndAlso _
388                                              Me.RegEngFile <> "" AndAlso _
389                                              Me.RegTranFile <> "")
390             End If
391         End With
392     End Sub
393
394     Private Sub OnToolBar_ButtonClick(ByVal sender As System.Object, ByVal e As System.
        Windows.Forms.ToolBarButtonClickEventArgs) Handles m_ToolBar.ButtonClick
```

```
395         'If e.Button = Me.m_TBarExit Then
396         '    Me.Close()
397         'End If
398         Debug.WriteLine("Buttons: " + e.Button.ToString + " " + Me.m_TBarStartModel.
        ToString)
399         Select Case Me.m_ToolBar.Buttons.IndexOf(e.Button)
400             Case 0
401                 ' Open All Files
402                 Trace.WriteLine("Open Input Button")
403                 If CAFE.OtherSettings.SingleFileDatasetInput Then
404                     Me.OpenMarketDataFile()
405                 Else
406                     Me.OpenVehiclesFile()
407                     Me.OpenEnginesFile()
408                     Me.OpenTransmissionsFile()
409                 End If
410                 Me.OpenTechnologyFile()
411                 Me.OpenParametersFile()
412                 Me.OpenScenariosFile()
413             Case 2
414                 ' Open Other Files
415                 Trace.WriteLine("Open Others Button")
416                 Me.OpenTechnologyFile()
417                 Me.OpenParametersFile()
418                 Me.OpenScenariosFile()
419             Case 4
420                 ' Open Options Window
421                 Trace.WriteLine("Options Button")
422                 m_MenuOptionsDialog_Click(sender, e)
423             Case 6
424                 ' Start Model
425                 Trace.WriteLine("Start Model Button")
426                 If CAFE.ModelRunning = False Then
427                     RunModel()
428                 End If
429             Case 7
430                 ' Stop Model/Reports
431                 If Me.m_TBarCancelModeling.Text = "Cancel Model" Then
432                     Trace.WriteLine("Cancel Model Button")
433                     If CAFE.ModelRunning Then
434                         CAFE.ModelRunning = False
435                         CAFE.ModelingCanceled = True
436                     End If
437                 Else
438                     Trace.WriteLine("Cancel Reports Button")
439                     Output.CancelReports()
440                 End If
441             Case 10
442                 ' Exit the model
443                 Trace.WriteLine("Exit Model Button")
444                 Me.Close()
445         End Select
446     End Sub
447
448     '''
449     ' Opens the registry key.
450     '
451     ' This function first checks if the registry key for the current version exists.  If it
        does,
452     ' then the current version registry key is returned, otherwise the key for the new
        version is
453     ' created and a check is made for the previous version's registry key.
454     ' If the previous version's registry entry exists, it is returned.  Otherwise, the
        previously
455     ' created current version registry key is returned.
456     '
457     Private Function OpenRegKey() As RegistryKey
458         Dim current As String
459         Dim previous As String
460         Dim ver As String = Application.ProductVersion
```

```
461
462         current = "Software\" + _
463                   Application.CompanyName + "\" + _
464                   Application.ProductName + "\Version " + _
465                   ver.Substring(0, ver.IndexOf("."))
466
467         previous = "Software\" + Application.CompanyName + "\" + Application.ProductName + "\1.0.0.0"
468
469         ' check if registry entry does not exist for current version
470         If IsNothing(Registry.CurrentUser.OpenSubKey(current)) Then
471             ' registry entry for current version does not exist, create it.
472             Registry.CurrentUser.CreateSubKey(current)
473
474             ' then check if there is an entry for the previous version (1.0.0.0)...
475             If Not IsNothing(Registry.CurrentUser.OpenSubKey(previous, True)) Then
476                 ' previous version (ver 1.0.0.0) entry exists, use it
477                 Return Registry.CurrentUser.OpenSubKey(previous, True)
478             End If
479         End If
480
481         ' if the current version reg key exists, return it; or if it does not exist but there was
482         ' no previous version reg key, return the newly created current version reg key
483         Return Registry.CurrentUser.OpenSubKey(current, True)
484     End Function
485
486
487 Main Menu Routines
488
489
490 Context Menu Routines
491
492
493 File Open and Input Routines
494
495
496     Private Sub RunModel()
497         If CAFE.ModelRunning Then
498             Exit Sub
499         End If
500
501         ' if any of the files are missing, exit...
502         If Me.NHTSA_VehiclesFileName = String.Empty OrElse _
503            Me.NHTSA_EnginesFileName = String.Empty OrElse _
504            Me.NHTSA_TransmissionsFileName = String.Empty OrElse _
505            Me.TechnologyFileName = String.Empty OrElse _
506            Me.ParametersFileName = String.Empty OrElse _
507            Me.ScenariosFileName = "" Then
508             '
509             MsgBox("One or more files were not loaded." & NewLine & NewLine & _
510                    "Please try loading the files again.", MsgBoxStyle.Exclamation, "Missing File(s)")
511             Exit Sub
512         End If
513
514         ' prompt for the output path
515         Dim outputPath As String = Output.GetOutputPath()
516         If outputPath = "" Then
517             MsgBox("Please select an output path.", MsgBoxStyle.Information, "Select an Output Path")
518             Return
519         Else
520             CAFE.OutputSettings.DefaultOutputPath = outputPath
521         End If
522
523         Dim rkey As RegistryKey = Me.OpenRegKey()
524
525         ' Record filenames to registry for the next time...
526         If CAFE.OtherSettings.SingleFileDatasetInput And Not IsNothing(Me.
```

```
    NHTSA_MarketDataFileName) Then
527             rkey.SetValue("MarketDataDatasetFileName", Me.NHTSA_MarketDataFileName)
528         Else
529             rkey.SetValue("VehiclesDatasetFileName", Me.NHTSA_VehiclesFileName)
530             rkey.SetValue("EnginesDatasetFileName", Me.NHTSA_EnginesFileName)
531             rkey.SetValue("TransmissionsDatasetFileName", Me.NHTSA_TransmissionsFileName)
532         End If
533         rkey.SetValue("TechnologyFileName", Me.TechnologyFileName)
534         rkey.SetValue("ParametersFileName", Me.ParametersFileName)
535         rkey.SetValue("ScenariosFileName", Me.ScenariosFileName)
536
537         ' Record option settings to registry for next time
538         ' Record technology settings to registry
539         rkey.SetValue("NRCPath", CInt(CAFE.PathSettings))
540         rkey.SetValue("NRCCost", CInt(CAFE.TechnologyCostEstimates))
541         rkey.SetValue("NRCFuelConsumption", CInt(CAFE.FuelConsumptionSavingsEstimates))
542
543         ' Record operating modes to registry
544         With CAFE.OperatingModes
545             rkey.SetValue("NoFines", .NoFines)
546             rkey.SetValue("LowCostFirst", CInt(.Approach))
547             rkey.SetValue("AutoToggleApproach", .AutoToggleApproach)
548             rkey.SetValue("BackFillTechnology", .BackFillTechnology)
549             rkey.SetValue("StageAnalysis", .StageAnalysis)
550             rkey.SetValue("MonteCarloAnalysis", .MonteCarloAnalysis)
551             rkey.SetValue("MonteCarloIterations", .MonteCarloIterations)
552             rkey.SetValue("NewEngineAuto", .CreateEngineIfCarsInvolved)
553             rkey.SetValue("NewEngineRC", .CreateEngineIfMultipleRC)
554             rkey.SetValue("NewEngineEnable", .CreateEngineIfTechEnableMissMatch)
555             rkey.SetValue("NewTranAuto", .CreateTransmissionIfCarsInvolved)
556             rkey.SetValue("NewTranRC", .CreateTransmissionIfMultipleRC)
557             rkey.SetValue("NewTranEnable", .CreateTransmissionIfTechEnableMissMatch)
558             rkey.SetValue("SkipAuto", .SkipIfAutoLightTruckMix)
559             rkey.SetValue("SkipRC", .SkipIfMultiRC)
560             rkey.SetValue("SkipEnable", .SkipIfEnableMismatch)
561             rkey.SetValue("ReserveTechs", .ReserveTechsIfInCompliance)
562         End With
563
564         ' Record output settings to registry
565         Dim rkey_os As RegistryKey = rkey.CreateSubKey("Output Settings")
566         With CAFE.OutputSettings
567             rkey_os.SetValue("DefaultOutputPath", .DefaultOutputPath)
568             rkey_os.SetValue("HideUnusedRCs", .HideUnusedRCs)
569             rkey_os.SetValue("GenIndustryReport", .GenerateIndustryReport)
570             rkey_os.SetValue("GenMfgsReport", .GenerateMfgsReport)
571             rkey_os.SetValue("GenVehiclesReport", .GenerateVehiclesReport)
572             rkey_os.SetValue("GenEffectsReport", .GenerateEffectsReport)
573             rkey_os.SetValue("GenEAReport", .GenerateEnvironmentalAssessmentReport)
574         End With
575
576         ' Record template settings to registry
577         Dim rkey_ts As RegistryKey = rkey.CreateSubKey("Template Settings")
578         With CAFE.TemplateSettings
579             rkey_ts.SetValue("UseTemplates", .UseTemplates)
580             rkey_ts.SetValue("DefaultTemplatePath", .DefaultTemplatePath)
581         End With
582
583         ' Record other settings
584         With CAFE.OtherSettings
585             rkey.SetValue("RunEffectsModel", .RunEffectsModel)
586             rkey.SetValue("NoStartupWarning", .NoStartupWarning)
587             rkey.SetValue("SingleFileDatasetInput", .SingleFileDatasetInput)
588         End With
589
590         ' close the registry key
591         rkey.Close()
592
593         m_WarningProvider.SetError(m_LabelDummy, "")        ' turn off warning (if it's on ↙
    -- harmless otherwise)
594
```

```
595         CAFE.ModelRunning = True                    ' set model running status to true
596         Me.ModelStartTime = DateTime.Now()          ' We're on the clock now...
597         m_TimerModel.Enabled = True
598         Me.m_TBarStartModel.Pushed = True
599         Me.m_TBarCancelModeling.Enabled = True
600         SetStatusBarStatus("Model running")
601         CAFE.PreProcess()                           ' pre-process the model
602
603         EnableProgressDisplay()
604
605         ' Monte Carlo or normal
606         If CAFE.OperatingModes.MonteCarloAnalysis Then
607             Dim MonteLogic As MonteCarloClass = New MonteCarloClass(CAFE.OperatingModes.
    MonteCarloIterations)
608         Else
609             CAFE.SummaryOutput = New SummaryInfoClass
610             Dim ModelLogic As New ModellingLogic    ' create a new ModellingLogic instance
611             ModelLogic.Begin()                      ' begin the model
612             CAFE.SummaryOutput.Close()
613         End If
614
615         CAFE.PostProcess()                          ' post-process the model
616         m_TimerModel.Enabled = False
617         CAFE.ModelRunning = False                   ' set model running status to false
618         DisableProgressDisplay()
619         Me.QuickExit = True
620         SetStatusBarStatus("Modeling complete")
621         Beep()
622         Me.m_TBarStartModel.Pushed = False
623         Me.m_TBarCancelModeling.Enabled = False
624
625         ' generate reports
626         If Not CAFE.OperatingModes.MonteCarloAnalysis Then
627             With CAFE.OutputSettings
628                 If .GenerateIndustryReport OrElse .GenerateMfgsReport OrElse _
629                    .GenerateVehiclesReport OrElse .GenerateEffectsReport OrElse _
630                    .GenerateEnvironmentalAssessmentReport Then
631                     '
632                     Me.GenerateReports(False, False)
633                 End If
634             End With
635         End If
636     End Sub
637
638     Private Sub OnOptionsMY(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles m_MenuOptionsMY.Click
639         Trace.WriteLine("Model Year Settings Menu")
640         Dim MYSettings As MYForm = New MYForm(Me, 0)
641         MYSettings.ShowDialog()
642     End Sub
643
644     Private Sub OnContextOptionsPathSelect(ByVal sender As Object, ByVal e As System.
    EventArgs) Handles m_ContextOptionsPaths.Select
645         Me.m_ContextOptionsPath1.Checked = (CAFE.PathSettings = CAFE.PathsEnum.Path1)
646         Me.m_ContextOptionsPath2.Checked = (CAFE.PathSettings = CAFE.PathsEnum.Path2)
647         Me.m_ContextOptionsPath3.Checked = (CAFE.PathSettings = CAFE.PathsEnum.Path3)
648     End Sub
649
650     Private Sub OnContextOptionsFuelSelect(ByVal sender As Object, ByVal e As System.
    EventArgs) Handles m_ContextOptionsFuel.Select
651         Me.m_ContextOptionsFuelLow.Checked = (CAFE.FuelConsumptionSavingsEstimates = CAFE.
    LowAverageHigh.Low)
652         Me.m_ContextOptionsFuelAverage.Checked = (CAFE.FuelConsumptionSavingsEstimates =
    CAFE.LowAverageHigh.Average)
653         Me.m_ContextOptionsFuelHigh.Checked = (CAFE.FuelConsumptionSavingsEstimates = CAFE.
    LowAverageHigh.High)
654     End Sub
655
656     Private Sub OnContextOptionsCostSelect(ByVal sender As Object, ByVal e As System.
    EventArgs) Handles m_ContextOptionsCost.Select
```

```
657         Me.m_ContextOptionsCostLow.Checked = (CAFE.TechnologyCostEstimates = CAFE.
        LowAverageHigh.Low)
658         Me.m_ContextOptionsCostAverage.Checked = (CAFE.TechnologyCostEstimates = CAFE.
        LowAverageHigh.Average)
659         Me.m_ContextOptionsCostHigh.Checked = (CAFE.TechnologyCostEstimates = CAFE.
        LowAverageHigh.High)
660     End Sub
661
662     Private Sub OnContextOptionsPath1(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_ContextOptionsPath1.Click
663         Me.OnOptionsPath1(sender, e)
664     End Sub
665
666     Private Sub OnContextOptionsPath2(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_ContextOptionsPath2.Click
667         Me.OnOptionsPath2(sender, e)
668     End Sub
669
670     Private Sub OnContextOptionsPath3(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_ContextOptionsPath3.Click
671         Me.OnOptionsPath3(sender, e)
672     End Sub
673
674     Private Sub OnContextOptionsFuelLow(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_ContextOptionsFuelLow.Click
675         Me.OnOptionsFuelLow(sender, e)
676     End Sub
677
678     Private Sub OnContextOptionsFuelAverage(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles m_ContextOptionsFuelAverage.Click
679         Me.OnOptionsFuelAverage(sender, e)
680     End Sub
681
682     Private Sub OnContextOptionsFuelHigh(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_ContextOptionsFuelHigh.Click
683         Me.OnOptionsFuelHigh(sender, e)
684     End Sub
685
686     Private Sub OnContextOptionsCostLow(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_ContextOptionsCostLow.Click
687         Me.OnOptionsCostLow(sender, e)
688     End Sub
689
690     Private Sub OnContextOptionsCostAverage(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles m_ContextOptionsCostAverage.Click
691         Me.OnOptionsCostAverage(sender, e)
692     End Sub
693
694     Private Sub OnContextOptionsCostHigh(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_ContextOptionsCostHigh.Click
695         Me.OnOptionsCostHigh(sender, e)
696     End Sub
697
698     Private Sub OnContextOptionsMY(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_ContextOptionsMY.Click
699         Me.OnOptionsMY(sender, e)
700     End Sub
701
702     Private Sub OnMainGenerateReports(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_MainGenerateReports.Click
703         Me.GenerateReports(True, True)
704     End Sub
705
706     '''
707     ' This method is used for calling the Output.GenerateReports() function and reporting
        any
708     ' errors occured during report generation.
709     '
710     Private Sub GenerateReports(ByVal SaveSettings As Boolean, ByVal GetPath As Boolean)
711         If SaveSettings Then
```

```
712             Dim rkey_os As RegistryKey = Me.OpenRegKey().CreateSubKey("Output Settings")
713             Dim rkey_ts As RegistryKey = Me.OpenRegKey().CreateSubKey("Template Settings")
714
715             ' Record output settings to registry
716             With CAFE.OutputSettings
717                 rkey_os.SetValue("DefaultOutputPath", .DefaultOutputPath)
718                 rkey_os.SetValue("HideUnusedRCs", .HideUnusedRCs)
719                 rkey_os.SetValue("GenIndustryReport", .GenerateIndustryReport)
720                 rkey_os.SetValue("GenMfgsReport", .GenerateMfgsReport)
721                 rkey_os.SetValue("GenVehiclesReport", .GenerateVehiclesReport)
722                 rkey_os.SetValue("GenEffectsReport", .GenerateEffectsReport)
723                 rkey_os.SetValue("GenEAReport", .GenerateEnvironmentalAssessmentReport)
724             End With
725
726             ' Record template settings to registry
727             With CAFE.TemplateSettings
728                 rkey_ts.SetValue("UseTemplates", .UseTemplates)
729                 rkey_ts.SetValue("DefaultTemplatePath", .DefaultTemplatePath)
730             End With
731
732             rkey_os.Close()
733             rkey_ts.Close()
734         End If
735
736         Dim NoErrors As Boolean          ' no report errors
737         Dim nErrors As Integer = 0
738         Dim Errors() As String
739
740         ' before running the reports...
741         ' disable the start model and exit buttons
742         Me.m_TBarStartModel.Enabled = False
743         Me.m_TBarExit.Enabled = False
744         ' and rename the cancel model button
745         Me.m_TBarCancelModeling.Text = "Cancel Reports"
746         Me.m_TBarCancelModeling.Enabled = True
747
748         Me.m_StatusBarModelTime.Text = ""
749         Me.SetStatusBarStatus("Generating reports.")
750         NoErrors = Output.GenerateReports(GetPath, nErrors, Errors)
751         Beep()
752
753         If NoErrors Then
754             Me.SetStatusBarStatus("Report generation complete.")
755         Else
756             Dim i As Integer
757             Dim ErrorString As String = ""
758
759             For i = 0 To nErrors - 1
760                 ErrorString += ((i + 1) & ".  " & Errors(i) & NewLine)
761             Next i
762
763             ' if there were any errors in the report, display the last error in the status ↙
    bar
764             Me.SetStatusBarStatus(Errors(nErrors - 1))
765             MsgBox("The following error(s) occured during report generation:" & NewLine & ↙
    NewLine & _
766                    ErrorString & NewLine & "Please try again.", _
767                    MsgBoxStyle.Information, "Report Errors")
768         End If
769
770         ' after running the reports...
771         ' re-enable the start model and exit buttons
772         Me.m_TBarStartModel.Enabled = True
773         Me.m_TBarExit.Enabled = True
774         ' and rename the cancel model button
775         Me.m_TBarCancelModeling.Text = "Cancel Model"
776         Me.m_TBarCancelModeling.Enabled = False
777     End Sub
778
779     Private Sub m_MainOpenLast_Click(ByVal sender As System.Object, ByVal e As System. ↙
```

```
    EventArgs) Handles m_MainOpenLast.Click
780         ' Open files using file names retrieved from the registry.
781         If CAFE.OtherSettings.SingleFileDatasetInput AndAlso Me.RegMarketDataFile <> ""
    Then
782             Me.OpenMarketDataFile(Me.RegMarketDataFile)
783         ElseIf Me.RegVehFile <> "" AndAlso RegEngFile <> "" AndAlso RegTranFile <> "" Then
784             Me.OpenVehiclesFile(Me.RegVehFile)
785             Me.OpenEnginesFile(Me.RegEngFile)
786             Me.OpenTransmissionsFile(Me.RegTranFile)
787         End If
788
789         If Me.RegTechFile <> "" AndAlso Me.RegParamFile <> "" Then
790             Me.OpenTechnologyFile(Me.RegTechFile)
791             Me.OpenParametersFile(Me.RegParamFile)
792         End If
793
794         If Me.RegScenFile <> "" Then
795             Me.OpenScenariosFile(Me.RegScenFile)
796         End If
797     End Sub
798
799     Private Sub m_MainOptionsNewEngine_Select(ByVal sender As Object, ByVal e As System.
    EventArgs) Handles m_MainOptionsNewEngine.Select
800         m_MainOptionsNewEngineAutoTruckMix.Enabled = Not CAFE.ModelRunning
801         m_MainOptionsNewEngineRCMix.Enabled = Not CAFE.ModelRunning
802         m_MainOptionsNewEngineEnableMix.Enabled = Not CAFE.ModelRunning
803         m_MainOptionsNewEngineAutoTruckMix.Checked = CAFE.OperatingModes.
    CreateEngineIfCarsInvolved
804         m_MainOptionsNewEngineRCMix.Checked = CAFE.OperatingModes.CreateEngineIfMultipleRC
805         m_MainOptionsNewEngineEnableMix.Checked = CAFE.OperatingModes.
    CreateEngineIfTechEnableMissMatch
806     End Sub
807
808     Private Sub m_MainOptionsNewEngineAutoTruckMix_Click(ByVal sender As Object, ByVal e As
     System.EventArgs) Handles m_MainOptionsNewEngineAutoTruckMix.Click
809         CAFE.OperatingModes.CreateEngineIfCarsInvolved = Not CAFE.OperatingModes.
    CreateEngineIfCarsInvolved
810     End Sub
811
812     Private Sub m_MainOptionsNewEngineRCMix_Click(ByVal sender As Object, ByVal e As System
    .EventArgs) Handles m_MainOptionsNewEngineRCMix.Click
813         CAFE.OperatingModes.CreateEngineIfMultipleRC = Not CAFE.OperatingModes.
    CreateEngineIfMultipleRC
814     End Sub
815
816     Private Sub m_MainOptionsNewEngineEnableMix_Click(ByVal sender As Object, ByVal e As
    System.EventArgs) Handles m_MainOptionsNewEngineEnableMix.Click
817         CAFE.OperatingModes.CreateEngineIfTechEnableMissMatch = Not CAFE.OperatingModes.
    CreateEngineIfTechEnableMissMatch
818     End Sub
819
820     Private Sub On_MainOptionsNewTrans_Select(ByVal sender As Object, ByVal e As System.
    EventArgs) Handles m_MainOptionsNewTrans.Select
821         m_MainOptionsNewTransAutoTruckMix.Enabled = Not CAFE.ModelRunning
822         m_MainOptionsNewTransRCMix.Enabled = Not CAFE.ModelRunning
823         m_MainOptionsNewTransEnableMix.Enabled = Not CAFE.ModelRunning
824         m_MainOptionsNewTransAutoTruckMix.Checked = CAFE.OperatingModes.
    CreateTransmissionIfCarsInvolved
825         m_MainOptionsNewTransRCMix.Checked = CAFE.OperatingModes.
    CreateTransmissionIfMultipleRC
826         m_MainOptionsNewTransEnableMix.Checked = CAFE.OperatingModes.
    CreateTransmissionIfTechEnableMissMatch
827     End Sub
828
829     Private Sub m_MainOptionsNewTransAutoTruckMix_Click(ByVal sender As Object, ByVal e As
    System.EventArgs) Handles m_MainOptionsNewTransAutoTruckMix.Click
830         CAFE.OperatingModes.CreateTransmissionIfCarsInvolved = Not CAFE.OperatingModes.
    CreateTransmissionIfCarsInvolved
831     End Sub
832
```

```
833     Private Sub m_MainOptionsNewTransRCMix_Click(ByVal sender As Object, ByVal e As System.
        EventArgs) Handles m_MainOptionsNewTransRCMix.Click
834         CAFE.OperatingModes.CreateTransmissionIfMultipleRC = Not CAFE.OperatingModes.
        CreateTransmissionIfMultipleRC
835     End Sub
836
837     Private Sub m_MainOptionsNewTransEnableMix_Click(ByVal sender As Object, ByVal e As
        System.EventArgs) Handles m_MainOptionsNewTransEnableMix.Click
838         CAFE.OperatingModes.CreateTransmissionIfTechEnableMissMatch = Not CAFE.
        OperatingModes.CreateTransmissionIfTechEnableMissMatch
839     End Sub
840
841     Private Sub m_MainRecentOptions_Click(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_MainRecentOptions.Click
842         If Me.RegEnteredOK Then
843             ' technology settings
844             CAFE.PathSettings = Me.RegPath
845             CAFE.TechnologyCostEstimates = Me.RegCost
846             CAFE.FuelConsumptionSavingsEstimates = Me.RegFC
847
848             ' operating modes
849             With CAFE.OperatingModes
850                 .NoFines = Me.RegNoFines
851                 .Approach = RegLowCostFirst
852                 .AutoToggleApproach = Me.RegAutoToggle
853                 .BackFillTechnology = Me.RegBackFill
854                 .StageAnalysis = Me.RegStageAnalysis
855                 .MonteCarloAnalysis = Me.RegMonteCarloAnalysis
856                 .MonteCarloIterations = Me.RegMonteCarloIterations
857                 .CreateEngineIfCarsInvolved = Me.RegNewEngAuto
858                 .CreateEngineIfMultipleRC = Me.RegNewEngRC
859                 .CreateEngineIfTechEnableMissMatch = Me.RegNewEngEnable
860                 .CreateTransmissionIfCarsInvolved = Me.RegNewTranAuto
861                 .CreateTransmissionIfMultipleRC = Me.RegNewTranRC
862                 .CreateTransmissionIfTechEnableMissMatch = Me.RegNewTranEnable
863                 .SkipIfAutoLightTruckMix = Me.RegSkipAuto
864                 .SkipIfMultiRC = Me.RegSkipRC
865                 .SkipIfEnableMismatch = Me.RegSkipEnable
866                 .ReserveTechsIfInCompliance = Me.RegReserveTechs
867             End With
868
869             ' output settings
870             With CAFE.OutputSettings
871                 .HideUnusedRCs = Me.RegOutputHideUnusedRcs
872                 .GenerateIndustryReport = Me.RegOutputInd
873                 .GenerateMfgsReport = Me.RegOutputMfgs
874                 .GenerateVehiclesReport = Me.RegOutputVehs
875                 .GenerateEffectsReport = Me.RegOutputEffs
876                 .GenerateEnvironmentalAssessmentReport = Me.RegOutputEAs
877             End With
878         End If
879     End Sub
880
881     Private Sub m_MainRecentFilesAndOptions_Click(ByVal sender As System.Object, ByVal e As
         System.EventArgs) Handles m_MainRecentFilesAndOptions.Click
882         Me.m_MainRecentOptions_Click(sender, e)
883         Me.m_MainOpenLast_Click(sender, e)
884     End Sub
885
886     Private Sub m_MainOptionsMostRecent_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles m_MainOptionsMostRecent.Click
887         m_MainRecentOptions_Click(sender, e)
888     End Sub
889
890     Private Sub m_MainLoad_N_Go_Click(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles m_MainLoad_N_Go.Click
891         m_MainRecentFilesAndOptions_Click(sender, e)
892         RunModel()
893     End Sub
894
```

```
895     Private Sub On_MenuOptions_Select(ByVal sender As Object, ByVal e As System.EventArgs) ↙
        Handles m_MenuOptions.Select
896         ' Use the checkboxes on the form as the source.
897         Me.m_MainOptionsLowCostFirst.Checked = (CAFE.OperatingModes.Approach = CAFE. ↙
        ApproachEnum.LowCostFirst)
898         Me.m_MainOptionsNoFines.Checked = CAFE.OperatingModes.NoFines
899         Me.m_MainOptionsBackFill.Checked = CAFE.OperatingModes.BackFillTechnology
900     End Sub
901
902     Private Sub m_MenuOptionsDialog_Click(ByVal sender As System.Object, ByVal e As System. ↙
        EventArgs) Handles m_MenuOptionsDialog.Click
903         Dim OptionsWindow As OptionsForm = New OptionsForm
904         OptionsWindow.ShowDialog()
905     End Sub
906
907     Private Sub On_MainOptionsLowCostFirst_Click(ByVal sender As Object, ByVal e As System. ↙
        EventArgs) Handles m_MainOptionsLowCostFirst.Click
908         If CAFE.OperatingModes.Approach = CAFE.ApproachEnum.LowCostFirst Then
909             CAFE.OperatingModes.Approach = CAFE.ApproachEnum.EconomicEfficiency
910         Else
911             CAFE.OperatingModes.Approach = CAFE.ApproachEnum.LowCostFirst
912         End If
913     End Sub
914
915     Private Sub On_MainOptionsNoFines_Click(ByVal sender As Object, ByVal e As System. ↙
        EventArgs) Handles m_MainOptionsNoFines.Click
916         CAFE.OperatingModes.NoFines = Not CAFE.OperatingModes.NoFines
917     End Sub
918
919     Private Sub On_MainOptionsBackFill_Click(ByVal sender As Object, ByVal e As System. ↙
        EventArgs) Handles m_MainOptionsBackFill.Click
920         CAFE.OperatingModes.BackFillTechnology = Not CAFE.OperatingModes.BackFillTechnology
921     End Sub
922
923     Private Sub m_CxBoxLowCostFirst_Click(ByVal sender As System.Object, ByVal e As System. ↙
        EventArgs)
924         On_MainOptionsLowCostFirst_Click(sender, e)
925     End Sub
926
927     Private Sub m_CxBoxNoFines_Click(ByVal sender As System.Object, ByVal e As System. ↙
        EventArgs)
928         On_MainOptionsNoFines_Click(sender, e)
929     End Sub
930
931     Private Sub m_CxBoxBackFillTech_Click(ByVal sender As System.Object, ByVal e As System. ↙
        EventArgs)
932         On_MainOptionsBackFill_Click(sender, e)
933     End Sub
934
935
936 StatusBar
937
938 Scenario, MY and Mfg ProgressBars
939
940 Model Time
941
942 CAFE and Fine ProgressBars
943
944
945     Private Sub m_MainAboutAbout_Click(ByVal sender As System.Object, ByVal e As System. ↙
        EventArgs) Handles m_MainAboutAbout.Click
946         Dim NewLine As String = Chr(13) + Chr(10)
947         If MessageBox.Show("U.S. Department of Transportation" & NewLine & NewLine & _
948         "Version:  " & VersionMajor.ToString & "." & VersionMinor.ToString & NewLine & _
949         "Date:      " & VersionDate & NewLine & NewLine & _
950         "Recommended Hardware:  1 GHz or faster processor with at least 512 MB RAM" & ↙
        NewLine & NewLine & _
951         "Required Software:  Microsoft® Windows XP, .NET Framework, and Office Excel 2003" ↙
        & NewLine & NewLine & _
952         "Contact:" & NewLine & NewLine & "Kevin Green" & NewLine & "U.S. Department of ↙
```

```
    Transportation" & NewLine & _
953         "Volpe National Transportation Systems Center" & NewLine & "55 Broadway" & NewLine
    & "Cambridge, MA  02142" & NewLine & NewLine & _
954         "Voice:   (617) 494-2106" & NewLine & "Fax:     (617) 494-3688" & NewLine & "E-Mail
    :  green@volpe.dot.gov", _
955         "CAFE Compliance and Effects Model", _
956               MessageBoxButtons.OK) = DialogResult.OK Then
957             ' Continue
958         End If
959     End Sub
960
961     Private Sub Warning()
962         'Dim NewLine As String = Chr(13) + Chr(10)
963         If MessageBox.Show("U.S. DEPARTMENT OF TRANSPORTATION" & NewLine & NewLine & _
964             "WARNINGS" & NewLine & NewLine & _
965             "1.  This software was developed for analysis by U.S. Department of
    Transportation staff" & NewLine & _
966             "of potential fuel economy requirements.  The software is not yet complete, and
     contains" & NewLine & _
967             "known and likely unknown errors.  Calculations reported by this software
    should thus be" & NewLine & _
968             "treated as provisional." & NewLine & NewLine & _
969             "2.  This software uses input files containing detailed information regarding
    vehicles" & NewLine & _
970             "manufactured for sale in the United States and creates output files containing
     similarly" & NewLine & _
971             "detailed information regarding such vehicles.  If input files containing
    information in" & NewLine & _
972             "any way (e.g., based on entitlement under 5 U.S.C 552 to confidential
    treatment)" & NewLine & _
973             "protected from disclosure to the public are used, some output files created by
     this" & NewLine & _
974             "software must also be protected from disclosure to the public." & NewLine &
    NewLine & _
975             "Do you understand these warnings?", _
976             "CAFE Compliance and Effects Model", _
977             MessageBoxButtons.YesNo) = DialogResult.Yes Then
978             ' Continue
979         Else
980             Application.Exit()
981         End If
982     End Sub
983
984 End Class
985
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This class stores data that will be outputed in reports
'''
Public MustInherit Class CafeStackClass

    Protected Enum MathOP
        None = 0
        Multiply = 1
        Divide = 2
        IntDivide = 3
        Add = 4
        Subtract = 5
    End Enum

    ' the scenario for this stack
    Protected Scenario As Integer

    ' the model year for this stack
    Public MY As Integer

    Public Valid As Boolean

    ' assumed total number of hours worked/year
    Public Const HoursPerYear As Double = 2080

    Public Sub New()
    End Sub

    Public MustOverride Function ToArray() As Object()

    Public Function GetStack() As Object()
        Return Me.ToArray()
    End Function

    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ''''''''''''''
    Protected Sub NanToZero(ByRef val As Double)
        If CAFE.NanToZero(val) = Double.Epsilon Then
            val = Double.NaN
        End If
    End Sub

    '''
    ' Sets all values to 0 or Double.Epsilon in a given array.
    '
    Protected Sub init(ByRef arr() As Double, ByVal val As Double)
        Dim i As Integer

        If val <> 0 And val <> Double.Epsilon Then
            val = 0
        End If

        For i = 0 To CType(arr, System.Array).Length - 1
            arr(i) = val
        Next i
    End Sub


 Math functions

End Class

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This class stores data per model year
'''
Public MustInherit Class CafeStacksClass

    Protected MyStack() As CafeStackClass
    Protected MySize As Integer
    Protected MinimumCapacity As Integer
```

```
70      Protected CapacityIncrement As Integer
71
72      Public Sub New()
73          Me.New(10, 10)
74      End Sub
75
76      Public Sub New(ByVal InitialCapacity As Integer)
77          Me.New(InitialCapacity, 10)
78      End Sub
79
80      Public Sub New(ByVal InitialCapacity As Integer, ByVal CapacityIncrement As Integer)
81          ReDim Me.MyStack(InitialCapacity - 1)
82          Me.MySize = 0
83          Me.MinimumCapacity = InitialCapacity
84          Me.CapacityIncrement = CapacityIncrement
85      End Sub
86
87      Protected Sub Copy(ByVal dst As CafeStacksClass)
88          Me.Copy(dst, Me)
89      End Sub
90
91      Protected Shared Sub Copy(ByRef dst As CafeStacksClass, ByRef src As CafeStacksClass)
92          dst.MySize = src.MySize
93          dst.MinimumCapacity = src.MinimumCapacity
94          dst.CapacityIncrement = src.CapacityIncrement
95      End Sub
96
97      Public MustOverride Function Push(ByVal csc As CafeStackClass) As Boolean
98
99      Public Function Pop() As CafeStackClass
100         Dim entry As CafeStackClass
101
102         Me.MySize -= 1
103         entry = Me.MyStack(Me.MySize)
104         Me.MyStack(Me.MySize) = Nothing
105
106         If Me.MySize < Me.MinimumCapacity And Me.MyStack.Length > Me.MinimumCapacity Then
107             ' MyStack array uses less than MinimumCapacity elements, but its size is
108             ' greater than MinimumCapacity--resize MyStack to be size of MinimumCapacity
109             ReDim Preserve Me.MyStack(Me.MinimumCapacity - 1)
110         ElseIf Me.MySize < Me.MyStack.Length - Me.CapacityIncrement Then
111             ' MyStack array has more than CapacityIncrement empty elements, so remove
112             ' the empty elements to save memory
113             ReDim Preserve Me.MyStack(Me.MyStack.Length - (Me.CapacityIncrement + 1))
114         End If
115
116         Return entry
117     End Function
118
119     Public Function Peek() As CafeStackClass
120         Return Me.MyStack(Me.MySize - 1)
121     End Function
122
123     Public Function Read() As CafeStackClass
124         Return Me.Peek()
125     End Function
126
127     Public Function Read(ByVal index As Integer) As CafeStackClass
128         If index < 0 Or index >= Me.MySize Then
129             ' index is out-of-bounds
130             Throw New System.IndexOutOfRangeException("Index must be greater than or equal " & _
131                                                       "to 0 and less than " & Me.MySize & ".")
132             Return Nothing
133         Else
134             Return Me.MyStack(index)
135         End If
136     End Function
137
```

```
138     Public Function Size() As Integer
139         Return Me.MySize
140     End Function
141
142 End Class
143
```

```
1  Imports System
2  Imports System.IO
3
4  Public Class C_CAFE_File
5      '   Inherits C_CAFE_Object
6
7      ' Constructor and destructor
8      Public Sub New()
9          'If IsNothing(m_ExcelApp) Then
10         '    m_ExcelApp = CForm.ExcelApp
11         '    m_Cform = CForm
12         'End If
13     End Sub
14     Protected Overrides Sub Finalize()
15         If IsNothing(Me.WorkBook) = False Then
16             Me.CloseFile(False)
17         End If
18     End Sub
19     Public Shared Function DoesFileExists(ByVal FileName As String) As Boolean
20         If IsNothing(FileName) = False AndAlso FileName <> String.Empty Then
21             Dim fi As New FileInfo(FileName)
22             DoesFileExists = fi.Exists
23             fi = Nothing
24         End If
25     End Function
26     Public Function IsCellEmpty(ByRef TheCell As Excel.Range) As Boolean
27         Dim rcode As Boolean
28         Dim TheVal As Object
29         Dim TheString As String
30         If IsArray(TheCell) Then
31             IsCellEmpty = False     ' whatever
32             Exit Function
33         End If
34         If IsNothing(TheCell) = False Then
35             If IsNothing(TheCell.Value) = False Then
36                 TheVal = TheCell.Value2
37
38                 TheString = CStr(TheVal)
39                 If System.String.Compare(TheString, "") <> 0 Then
40                     ' Cell exists but has something in it
41                     rcode = False
42                 Else
43                     ' cell exists but is empty
44                     rcode = True
45                 End If
46             Else
47                 ' Cell exists but the value is NOTHING
48                 rcode = True
49             End If
50         Else
51             ' Cell doesn't exist
52             rcode = True
53         End If
54         IsCellEmpty = rcode
55     End Function
56     Public Function IsWhiteSpace(ByRef wStr As String) As Boolean
57         Dim idx As Integer
58         Dim rCode As Boolean
59         rCode = True
60         If IsNothing(wStr) Then
61             ' For empty string, we'll return a 'TRUE' (at least for now)
62             rCode = True
63         ElseIf wStr.Length > 0 Then
64             For idx = 0 To wStr.Length - 1
65                 If Char.IsWhiteSpace(wStr, idx) = False Then
66                     rCode = False
67                     Exit For
68                 End If
69             Next idx
70         Else
```

```
138                     Me.WorkBook = ExcelApp.Workbooks.Open(CompleteFileName, , ReadOnlyFlag) ↙
        ' <---
139                     'Me.WorkBook = m_Cform.ExcelApp.Workbooks.Open(CompleteFileName, ,      ↙
    ReadOnlyFlag)
140                     Me.WorkBook.Application.Visible = False
141                     Me.WorkBook.Application.DisplayAlerts = False
142                     Me.WorkBook.Application.Calculation = Excel.XlCalculation.              ↙
    xlCalculationManual
143                     Me.WorkBook.Application.UserControl = False
144                 End If
145             Catch ex As Exception
146                 MsgBox("Could not open input file: " & ex.Message, MsgBoxStyle.Critical,    ↙
    "Cafe Reform")
147                 Return False
148             End Try
149         Else
150             ' File isn't readonly
151             If FileExists = True Then
152                 ' The file already exists.  WE need to delete it.
153                 Dim ffile As New FileInfo(CompleteFileName)
154                 Try
155                     fFile.Delete()
156                     FileExists = False
157                 Catch e As Exception
158                     MsgBox("Exception: " & e.ToString & " while trying to remove file: " &  ↙
    CompleteFileName, MsgBoxStyle.Exclamation)
159                     FileExists = True
160                 End Try
161                 If FileExists Then
162                     Return False     ' It was probably already opened
163                 End If
164             End If
165             ' Ok, here we are to open a new file.
166             ' Be sure the folder is there, if not, create it and then create the file
167             If PathExists = False Then
168                 fDir.CreateDirectory(Me.FilePath)
169             End If
170             '            Me.WorkBook = ExcelApp.Workbooks.Add
171             ExcelApp.Application.DisplayAlerts = False
172             Me.WorkBook = ExcelApp.Workbooks.Add
173             Me.WorkBook.Application.DisplayAlerts = False
174
175             If CAFE.EncryptionSettings.EncryptOutputFiles Then
176                 ' encrypt the saved files
177                 Me.SecureSave()
178             Else
179                 Me.WorkBook.SaveAs(CompleteFileName)
180             End If
181         End If
182
183         Return True
184     End Function
185     Public Overridable Sub CloseFile(ByVal SaveAnyChanges As Boolean)
186         If IsNothing(Me.WorkBook) = False Then
187             If Me.ReadOnlyFlag Then
188                 Me.WorkBook.Close(SaveChanges:=False)
189             Else
190                 Me.WorkBook.Close(SaveAnyChanges)
191             End If
192             Me.WorkBook = Nothing
193         End If
194     End Sub
195     Public Sub Activate()
196         ' Activates this workbook
197         If IsNothing(Me.WorkBook) = False Then
198             CType(Me.WorkBook, Excel.WorkbookClass).Activate()
199         End If
200     End Sub
201     Public Function GetWorkbookObject() As Excel.Workbook
202         Return Me.WorkBook
```

```
203     End Function
204     Public Function SetSheet(ByVal SheetNum As Integer) As Excel.Worksheet
205         Dim AC As Excel.Worksheet
206         Me.Activate()
207         If IsNothing(Me.WorkBook) = False Then
208             AC = Me.WorkBook.Worksheets(SheetNum)
209         Else
210             AC = Nothing
211         End If
212         SetSheet = AC
213     End Function
214     Public Function GetSheet(ByVal SheetNum As Integer) As Excel.Worksheet
215         If Not IsNothing(Me.WorkBook) Then
216             Return Me.WorkBook.Worksheets(SheetNum)
217         End If
218
219         Return Nothing
220     End Function
221     Public Sub SecureSave()
222         Me.WorkBook.SetPasswordEncryptionOptions(CAFE.EncryptionSettings.EncryptionProvider ↙
        ' _
223                                                  "RC4", _
224                                                  CAFE.EncryptionSettings. ↙
        EncryptionKeyLength, _
225                                                  CAFE.EncryptionSettings. ↙
        EncryptDocumentProperties)
226         '
227         Me.WorkBook.SaveAs(Me.GetFileNameAndPath(), _
228                            Excel.XlFileFormat.xlWorkbookNormal, _
229                            CAFE.EncryptionSettings.OutputPasswordToOpen, _
230                            CAFE.EncryptionSettings.OutputPasswordToModify, False, False)
231     End Sub
232     Public Sub SecureOpen(ByVal Password As String)
233         Me.WorkBook = ExcelApp.Workbooks.Open(Me.GetFileNameAndPath(), , Me.ReadOnlyFlag, , ↙
         Password, )
234     End Sub
235
236     ' Utility Methods
237     Private Function FileExists(ByVal fName As String) As Boolean
238         Dim ff As System.IO.File
239         FileExists = ff.Exists(fName)
240     End Function
241     Private Function FileExists() As Boolean
242         Return Me.FileExists(Me.GetFileNameAndPath())
243     End Function
244     Public Function GetFileName() As String
245         Return Me.FileName
246     End Function
247     Public Function GetFilePath() As String
248         Return Me.FilePath
249     End Function
250     Public Function GetFileNameAndPath() As String
251         Return Me.FilePath + "\" + Me.FileName
252     End Function
253     Public Sub SplitFileName(ByVal WholeFileName As String, ByRef RtnPathName As String, ↙
        ByRef RtnFileName As String)
254         Dim len As Integer
255         Dim li As Integer
256
257         len = WholeFileName.Length
258         li = WholeFileName.LastIndexOf("\")
259         RtnPathName = WholeFileName.Remove(li, len - li)
260         RtnFileName = WholeFileName.Remove(0, li + 1)
261
262     End Sub
263
264     ' Data members
265     Protected FileName As String
266     Protected FilePath As String
267     Protected ReadOnlyFlag As Boolean
```

```
268     Protected WorkBook As Excel.Workbook
269
270 End Class
271
272 Public Class C_Cafe_InFile
273     Inherits C_CAFE_File
274
275     Public Sub New()
276         MyBase.New()
277     End Sub
278     Protected Overrides Sub Finalize()
279         If IsNothing(Me.WorkBook) = False Then
280             Me.CloseFile()
281         End If
282         MyBase.Finalize()
283     End Sub
284     Public Overloads Function OpenFile(ByVal CompleteFileName As String, Optional ByVal
        WhichRawFile As Integer = -1) As Boolean
285         ' Open the file as readonly using base-class's open method
286         Return MyBase.OpenFile(CompleteFileName, True, WhichRawFile)
287     End Function
288     Public Overloads Sub CloseFile()
289         MyBase.CloseFile(False)
290     End Sub
291     Public Function GetNumRows() As Integer
292         CType(Me.WorkBook, Excel.WorkbookClass).Activate()
293         Return Me.GetNumRows(Me.WorkBook.ActiveSheet)
294     End Function
295     Public Function GetNumRows(ByVal WorkSheet As Excel.Worksheet) As Integer
296         Dim r As Integer = 0
297         Dim InData(,) As Object = WorkSheet.Range("A:A").Resize.Value
298
299         Do While IsNothing(InData(r + 1, 1)) = False
300             r = r + 1
301         Loop
302
303         Return r
304     End Function
305     Public Function GetNumColumns(ByVal WhichRow As Integer, _
306                                   Optional ByVal ScanBackwards As Boolean = False) As
        Integer
307         '
308         ' WhichRow is row number 1 or 2 used to determine column count
309         CType(Me.WorkBook, Excel.WorkbookClass).Activate()
310         Return Me.GetNumColumns(Me.WorkBook.ActiveSheet, WhichRow, ScanBackwards)
311     End Function
312     Public Function GetNumColumns(ByVal WorkSheet As Excel.Worksheet, _
313                                   ByVal WhichRow As Integer, _
314                                   Optional ByVal ScanBackwards As Boolean = False) As
        Integer
315         '
316         ' WhichRow is row number 1 or 2 used to determine column count
317         Dim c As Integer = 0
318         Dim cOff As Integer = 1
319         Dim increment As Integer = 1
320         Dim InData(,) As Object
321
322         If WhichRow = 1 Then
323             InData = WorkSheet.Range("1:1").Resize.Value
324         Else
325             InData = WorkSheet.Range("2:2").Resize.Value
326         End If
327
328         If ScanBackwards Then
329             c = 256
330             cOff = 0
331             increment = -1
332         End If
333
334         Do While (IsNothing(InData(1, c + cOff)) = (increment = -1))
```

```
335             c += increment
336         Loop
337
338         Return c
339     End Function
340
341
342     ' Data Properties
343     'Protected ColumnHeaderData(,) As Object
344 End Class
345
346 Public Class C_CAFE_OutFile
347     Inherits C_CAFE_File
348     Public Sub New(ByRef CForm As CafeForm, ByVal CompleteFileName As String)
349         MyBase.New()
350         Me.OpenFile(CompleteFileName)
351     End Sub
352     Protected Overrides Sub Finalize()
353         If IsNothing(Me.WorkBook) = False Then
354             MyBase.CloseFile(False)          ' If closing from destructor, then don't save
    changes
355         End If
356         MyBase.Finalize()
357     End Sub
358     Public Overloads Function OpenFile(ByVal CompleteFileName As String) As Boolean
359         ' Use base class to open writable file
360         Return MyBase.OpenFile(CompleteFileName, False, -1)
361     End Function
362     Public Overloads Sub CloseFile()
363         MyBase.CloseFile(True)
364     End Sub
365 End Class
366
```

```
1  Public Class CheckboxArrayClass
2      Inherits System.Collections.CollectionBase
3
4      Private ReadOnly HostForm As System.Windows.Forms.GroupBox
5      Private RowIdx As Integer
6      Private ColIdx As Integer
7      Private Orientation As Integer = 0      ' 0 for vertical, 1 for horizontal
8
9      ' This field is used to signal that the maximum number of rows and cols
10     ' has been exceeded, thus no more checkboxes can be displayed on screen.
11     ' If this field is true, a message box should be displayed on form_load
12     ' of parent prompting the user.
13     Public RowsColsExceeded As Boolean
14
15     Public AllChecked As Boolean = False
16     Public AllUnChecked As Boolean = False
17
18     ' Get property at index idx
19     Default Public ReadOnly Property Item(ByVal idx As Integer) As System.Windows.Forms.
       CheckBox
20         Get
21             Return CType(Me.List.Item(idx), System.Windows.Forms.CheckBox)
22         End Get
23     End Property
24
25     Public Sub New(ByVal host As System.Windows.Forms.GroupBox)
26         HostForm = host
27         RowIdx = 0
28         ColIdx = 0
29     End Sub
30
31     Public Function AddNewCheckbox(ByVal year As String) As System.Windows.Forms.CheckBox
32         ' Create a new instance of the CheckBox class.
33         Dim ChkBox As New System.Windows.Forms.CheckBox
34
35         ' Add the checkbox to the array and host's controls.
36         Me.List.Add(ChkBox)
37         HostForm.Controls.Add(ChkBox)
38
39         ' Set intial properties.
40         If Me.Orientation = 0 Then
41             ' vertical orientation
42             ChkBox.Top = RowIdx * 24 + 24
43             ChkBox.Left = ColIdx * 85 + 16
44         Else
45             ' horizontal orientation
46             ChkBox.Top = ColIdx * 24 + 24
47             ChkBox.Left = RowIdx * 85 + 16
48         End If
49
50         ChkBox.Width = 50
51         ChkBox.Tag = year
52         ChkBox.Text = year
53         ChkBox.Checked = CAFE.ModelYearsSelected(Count - 1)
54         ChkBox.BringToFront()
55
56         AddHandler chkbox.Click, AddressOf ClickHandler
57         AddHandler chkbox.MouseEnter, AddressOf MouseEnter
58         AddHandler chkbox.MouseLeave, AddressOf MouseLeave
59
60         ' increment row index
61         RowIdx += 1
62
63         ' if RowIdx >= 6, reset row to 0 and move on to the next col
64         If RowIdx >= 6 Then
65             RowIdx = 0
66             ColIdx += 1
67         End If
68
69         ' if ColIdx >= 5, then we have reached the bounds of the form
```

```
70          ' and cannot add any more checkboxes; display message
71          If ColIdx >= 5 Then
72              RowsColsExceeded = True
73          End If
74
75          Return chkbox
76      End Function
77
78      Public Sub Remove()
79          ' Check to be sure there is a button to remove.
80          If Me.Count > 0 Then
81              HostForm.Controls.Remove(Me(Me.Count - 1))
82              Me.List.RemoveAt(Me.Count - 1)
83          End If
84      End Sub
85
86      Public Sub ClickHandler(ByVal sender As Object, ByVal e As System.EventArgs)
87          'Dim i As Integer
88
89          'AllChecked = True
90
91          'For i = 0 To Count - 1
92          '    If Me(i).Checked = False Then
93          '        AllChecked = False
94          '        Exit For
95          '    End If
96          'Next i
97
98          'AllUnChecked = True
99
100         'For i = 0 To Count - 1
101         '    If Me(i).Checked = True Then
102         '        AllUnChecked = False
103         '        Exit For
104         '    End If
105         'Next i
106         'MessageBox.Show("you have clicked checkbox " & _
107         '                CType(CType(sender, System.Windows.Forms.CheckBox).Tag, String))
108     End Sub
109
110     ' Handler for mouse enter
111     Public Sub MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
112         ' CType(sender, System.Windows.Forms.CheckBox).BackColor = Control.DefaultBackColor
113     End Sub
114
115     ' Handler for mouse leave
116     Public Sub MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
117         ' CType(sender, System.Windows.Forms.CheckBox).BackColor = Control.DefaultBackColor
118     End Sub
119
120     Public Sub ChangeOrientation()
121         Dim i As Integer
122         Dim ri, ci As Integer
123
124         Me.Orientation = 1 - Me.Orientation
125
126         For i = 0 To Me.Count - 1
127             If Me.Orientation = 0 Then
128                 ' vertical orientation
129                 Me(i).Top = ri * 24 + 24
130                 Me(i).Left = ci * 85 + 16
131             Else
132                 ' horizontal orientation
133                 Me(i).Top = ci * 24 + 24
134                 Me(i).Left = ri * 85 + 16
135             End If
136
137             ri += 1
138
139             If ri >= 6 - Me.Orientation Then
```

```
140                 ri = 0
141                 ci += 1
142             End If
143         Next i
144     End Sub
145
146     Public Sub SetOrientation(ByVal NewOrientation As Integer)
147         ' if the new orientation is different from current orientation...
148         If NewOrientation <> Orientation Then
149             ' then change the current orientation
150             Me.ChangeOrientation()
151         End If
152         ' otherwise do nothing
153     End Sub
154
155     Public Function GetOrientation() As Integer
156         Return Me.Orientation
157     End Function
158 End Class
159
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This class stores data that will be outputed in reports
'''
Public Class EAStackClass
    Inherits CafeStackClass

    ' arrays are repsented as arr(MY)(CY)
    Public FC()() As Double
    Public CO()() As Double
    Public VOC()() As Double
    Public NOX()() As Double
    Public SO2()() As Double
    Public PM()() As Double
    Public CO2()() As Double

    Public nMY As Integer
    Public nCY As Integer

    Public Sub New()
        MyBase.New()
    End Sub

    Public Sub New(ByVal nMY As Integer, ByVal nCY As Integer)
        Me.New()
        Me.CreateStack(nMY, nCY)
    End Sub

    Public Sub CreateStack(ByVal nMY As Integer, ByVal nCY As Integer)
        Dim i, j As Integer

        Me.nMY = nMY
        Me.nCY = nCY

        ReDim Me.FC(Me.nMY - 1)
        ReDim Me.CO(Me.nMY - 1)
        ReDim Me.VOC(Me.nMY - 1)
        ReDim Me.NOX(Me.nMY - 1)
        ReDim Me.SO2(Me.nMY - 1)
        ReDim Me.PM(Me.nMY - 1)
        ReDim Me.CO2(Me.nMY - 1)

        For i = 0 To Me.nMY - 1
            ReDim Me.FC(i)(Me.nCY - 1)
            ReDim Me.CO(i)(Me.nCY - 1)
            ReDim Me.VOC(i)(Me.nCY - 1)
            ReDim Me.NOX(i)(Me.nCY - 1)
            ReDim Me.SO2(i)(Me.nCY - 1)
            ReDim Me.PM(i)(Me.nCY - 1)
            ReDim Me.CO2(i)(Me.nCY - 1)
        Next i
    End Sub

    Public Sub Copy(ByRef dst As EAStackClass)
        Me.Copy(dst, Me)
    End Sub

    Public Shared Sub Copy(ByRef dst As EAStackClass, ByRef src As EAStackClass)
        Dim i, j As Integer

        dst = New EAStackClass(src.nMY, src.nCY)

        For i = 0 To dst.nMY - 1
            For j = 0 To dst.nCY - 1
                dst.FC(i)(j) = src.FC(i)(j)
                dst.CO(i)(j) = src.CO(i)(j)
                dst.VOC(i)(j) = src.VOC(i)(j)
                dst.NOX(i)(j) = src.NOX(i)(j)
                dst.SO2(i)(j) = src.SO2(i)(j)
                dst.PM(i)(j) = src.PM(i)(j)
                dst.CO2(i)(j) = src.CO2(i)(j)
```

```
71              Next j
72          Next i
73      End Sub
74
75      Public Overrides Function ToArray() As Object()
76          Dim size As Integer = 7
77          Dim stack(size - 1) As Object
78
79          stack(0) = Me.FC
80          stack(1) = Me.CO
81          stack(2) = Me.VOC
82          stack(3) = Me.NOX
83          stack(4) = Me.SO2
84          stack(5) = Me.PM
85          stack(6) = Me.CO2
86
87          Return stack
88      End Function
89
90  End Class
91
92
93  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
94  ' This class stores data per model year
95  '''
96  Public Class EAsStackClass
97      Inherits CafeStacksClass
98
99      Public Overrides Function Push(ByVal csc As CafeStackClass) As Boolean
100         ' create and copy the new EAStackClass
101         Dim entry As EAStackClass
102         EAStackClass.Copy(entry, csc)
103         '
104         ' add it to the stack
105         Me.MyStack(Me.MySize) = entry
106         Me.MySize += 1
107
108         If Me.MySize >= Me.MyStack.Length Then
109             ' array at max capacity; expand the array
110             ReDim Preserve Me.MyStack(Me.MySize + (Me.CapacityIncrement - 1))
111         End If
112
113         Return True
114     End Function
115
116 End Class
117
118
119
120
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This class stores data that will be outputed in reports
'''
Public Class EffectStackClass
    Inherits CafeStackClass

    Private Const ecLen As Integer = 3
    Private Const emLen As Integer = 9
    Private Const sftLen As Integer = 18

    ' effects data at the specific model year
    '
    ' Energy Consumption (Gas, Diesel, Total; all arrs are of len 3)
    Public EC_LifetimeFuelConsumption(ecLen - 1) As Double
    Public EC_Sales(ecLen - 1) As Double
    Public EC_LifetimeVMT(ecLen - 1) As Double
    Public EC_AverageFuelEconomy(ecLen - 1) As Double
    ' Emissions (LDDV, ..., HDGV2b, Total; all arrs are of len 9)
    Public Em_CO2(emLen - 1) As Double
    Public Em_CO(emLen - 1) As Double
    Public Em_VOC(emLen - 1) As Double
    Public Em_NOX(emLen - 1) As Double
    Public Em_PM(emLen - 1) As Double
    Public Em_SOX(emLen - 1) As Double
    Public Em_Sales(emLen - 1) As Double
    Public Em_LifetimeVMT(emLen - 1) As Double
    Public Em_AverageFuelEconomy(emLen - 1) As Double
    ' Safety (Very Small Cars, ..., Large Van, Total; all arrs are of len 18)
    Public Sft_TotalLifetimeFatalities(sftLen - 1) As Double
    Public Sft_Sales(sftLen - 1) As Double
    'Public Sft_LifetimeVMT(sftLen - 1) As Double
    Public Sft_AverageCurbWeight(sftLen - 1) As Double
    ' Undiscounted Owner and Societal Costs (15 total)
    Public TotalLifetimePretaxFuelExpenditures As Double
    Public FuelTaxRevenues As Double
    Public TravelValue As Double
    Public RefuelingTimeValue As Double
    Public PetroleumMarketExternalities As Double
    Public CongestionCosts As Double
    Public NoiseCosts As Double
    Public AccidentCosts As Double
    Public CO2 As Double
    Public CO As Double
    Public VOC As Double
    Public NOX As Double
    Public PM As Double
    Public SOX As Double
    Public MonetizedFatalities As Double
    ' Discounted Owner and Societal Costs (15 total)
    Public DiscTotalLifetimePretaxFuelExpenditures As Double
    Public DiscFuelTaxRevenues As Double
    Public DiscTravelValue As Double
    Public DiscRefuelingTimeValue As Double
    Public DiscPetroleumMarketExternalities As Double
    Public DiscCongestionCosts As Double
    Public DiscNoiseCosts As Double
    Public DiscAccidentCosts As Double
    Public DiscCO2 As Double
    Public DiscCO As Double
    Public DiscVOC As Double
    Public DiscNOX As Double
    Public DiscPM As Double
    Public DiscSOX As Double
    Public DiscMonetizedFatalities As Double

    Public Sub New()
        MyBase.New()
        Me.init(Double.Epsilon)
    End Sub
```

```
70
71      '''
72      ' Switch = 0, 1, or 2
73      '        0 = save regular values
74      '        1 = save absolute delta values
75      '        2 = save percent delta values
76      '
77      Public Sub New(ByVal Scenario As Integer, _
78                     ByVal MY As Integer, _
79                     Optional ByVal Switch As Integer = 0)
80          '
81          Me.New()
82          Me.CreateStack(Scenario, MY, Switch)
83      End Sub
84
85      '''
86      ' Sets all values to val (0 or Double.Epsilon), where epsilon indicates that these values ↙
         have
87      ' not been calculated yet.
88      '
89      Private Overloads Sub init(ByVal val As Double)
90          If val <> 0 And val <> Double.Epsilon Then
91              val = 0
92          End If
93
94          Me.init(Me.EC_LifetimeFuelConsumption, val)
95          Me.init(Me.EC_Sales, val)
96          Me.init(Me.EC_LifetimeVMT, val)
97          Me.init(Me.EC_AverageFuelEconomy, val)
98
99          Me.init(Me.Em_CO2, val)
100         Me.init(Me.Em_CO, val)
101         Me.init(Me.Em_VOC, val)
102         Me.init(Me.Em_NOX, val)
103         Me.init(Me.Em_PM, val)
104         Me.init(Me.Em_SOX, val)
105         Me.init(Me.Em_Sales, val)
106         Me.init(Me.Em_LifetimeVMT, val)
107         Me.init(Me.Em_AverageFuelEconomy, val)
108
109         If CAFE.SafetyIncluded Then
110             Me.init(Me.Sft_TotalLifetimeFatalities, val)
111             Me.init(Me.Sft_Sales, val)
112             'Me.init(Me.Sft_LifetimeVMT, val)
113             Me.init(Me.Sft_AverageCurbWeight, val)
114         End If
115
116         Me.TotalLifetimePretaxFuelExpenditures = val
117         Me.FuelTaxRevenues = val
118         Me.TravelValue = val
119         Me.RefuelingTimeValue = val
120         Me.PetroleumMarketExternalities = val
121         Me.CongestionCosts = val
122         Me.NoiseCosts = val
123         Me.AccidentCosts = val
124         Me.CO2 = val
125         Me.CO = val
126         Me.VOC = val
127         Me.NOX = val
128         Me.PM = val
129         Me.SOX = val
130         Me.MonetizedFatalities = val
131
132         Me.DiscTotalLifetimePretaxFuelExpenditures = val
133         Me.DiscFuelTaxRevenues = val
134         Me.DiscTravelValue = val
135         Me.DiscRefuelingTimeValue = val
136         Me.DiscPetroleumMarketExternalities = val
137         Me.DiscCongestionCosts = val
```

```
138         Me.DiscNoiseCosts = val
139         Me.DiscAccidentCosts = val
140         Me.DiscCO2 = val
141         Me.DiscCO = val
142         Me.DiscVOC = val
143         Me.DiscNOX = val
144         Me.DiscPM = val
145         Me.DiscSOX = val
146         Me.DiscMonetizedFatalities = val
147     End Sub
148
149     '''
150     ' Switch = 0, 1, or 2
151     '         0 = save regular values
152     '         1 = save absolute delta values
153     '         2 = save percent delta values
154     '
155     Public Shadows Function CreateStack(ByVal Scenario As Integer, _
156                                         ByVal MY As Integer, _
157                                         Optional ByVal Switch As Integer = 0) As
        EffectStackClass
158         '
159         Me.Scenario = Scenario
160         Me.MY = MY
161
162         Select Case Switch
163             Case 0      ' regular vals
164                 Me.CreateStackReg(MY)
165             Case 1      ' absolute delta vals
166                 Me.CreateStackDelta(Scenario, MY, ManufacturerStackClass.MathOP.Subtract)
167             Case 2      ' % delta vals
168                 Me.CreateStackDelta(Scenario, MY, ManufacturerStackClass.MathOP.Divide)
169         End Select
170
171         Return Me
172     End Function
173
174     Private Sub CreateStackReg(ByVal MY As Integer)
175         Dim i, j As Integer
176         Dim M6Classes() As AggregateVehiclesClass.MyData
177         Dim SftClasses() As AggregateVehiclesClass.MyData
178         Dim Totals As AggregateVehiclesClass.MyData
179         Dim MYIdx As Integer
180
181         ' initialize to 0.0 for regular stack vals
182         init(0.0)
183
184         MYIdx = CAFE.GMYI(MY)
185         M6Classes = CAFE.ScenarioAggregateData(MYIdx).GetMobile6Classes()
186         SftClasses = CAFE.ScenarioAggregateData(MYIdx).GetSafetyClasses()
187         Totals = CAFE.ScenarioAggregateData(MYIdx).GetTotals()
188
189         For i = 0 To ecLen - 1
190             Me.EC_Sales(i) = M6Classes(i + 9).Sales
191             Me.EC_LifetimeVMT(i) = Me.SumArray(M6Classes(i + 9).VMT)
192             Me.EC_AverageFuelEconomy(i) = M6Classes(i + 9).FuelEconomy
193         Next i
194
195         For i = 0 To emLen - 2
196             Me.Em_Sales(i) = M6Classes(i + 1).Sales
197             Me.Em_LifetimeVMT(i) = Me.SumArray(M6Classes(i + 1).VMT)
198             Me.Em_AverageFuelEconomy(i) = M6Classes(i + 1).FuelEconomy
199         Next i
200         Me.Em_Sales(i) = M6Classes(11).Sales
201         Me.Em_LifetimeVMT(i) = Me.SumArray(M6Classes(11).VMT)
202         Me.Em_AverageFuelEconomy(i) = M6Classes(11).FuelEconomy
203
204         If CAFE.SafetyIncluded Then
205             For i = 0 To sftLen - 2
```

```
206                 Me.Sft_Sales(i) = SftClasses(i + 1).Sales
207                 'Me.Sft_LifetimeVMT(i) = Me.SumArray(SftClasses(i + 1).VMT)
208                 Me.Sft_AverageCurbWeight(i) = SftClasses(i + 1).CurbWeight
209             Next i
210             Me.Sft_Sales(i) = SftClasses(18).Sales
211             'Me.Sft_LifetimeVMT(i) = Me.SumArray(SftClasses(18).VMT)
212             Me.Sft_AverageCurbWeight(i) = SftClasses(18).CurbWeight
213         End If
214
215         ' save owner and societal costs
216         ' undiscounted
217         Me.TotalLifetimePretaxFuelExpenditures = Me.SumArray(Totals.PreTaxFuel)
218         Me.FuelTaxRevenues = Me.SumArray(Totals.FuelTax)
219         Me.TravelValue = Me.SumArray(Totals.DriveSurplus)
220         Me.RefuelingTimeValue = Me.SumArray(Totals.RefuelSurplus)
221         ' discounted
222         Me.DiscTotalLifetimePretaxFuelExpenditures = Me.SumArray(Totals.DiscPreTaxFuel)
223         Me.DiscFuelTaxRevenues = Me.SumArray(Totals.DiscFuelTax)
224         Me.DiscTravelValue = Me.SumArray(Totals.DiscDriveSurplus)
225         Me.DiscRefuelingTimeValue = Me.SumArray(Totals.DiscRefuelSurplus)
226     End Sub
227
228     Private Overloads Sub CreateStackDelta(ByVal Scenario As Integer, ByVal MY As Integer,
        ByVal OP As MathOP)
229         ' If we're doing Monte Carlo, then we don't need to generate deltas
230         If CAFE.OperatingModes.MonteCarloAnalysis Then
231             Return
232         End If
233         ' check if the current scenario is 0; init and exit if so
234         If Scenario = 0 Then
235             init(Double.Epsilon)
236             Return
237         End If
238
239         ' the 0-th scenario is the baseline scenario
240         Dim BASELINE As Integer = 0
241
242         Dim i As Integer
243         Dim ThisEff As EffectStackClass
244         Dim BaselineEff As EffectStackClass
245
246         ' get the baseline values for this effect from the stack
247         For i = 0 To CAFE.EffectsStack(BASELINE).Size() - 1 Step 3
248             If CAFE.EffectsStack(BASELINE).Read(i).MY = MY Then
249                 BaselineEff = CAFE.EffectsStack(BASELINE).Read(i)
250                 Exit For
251             End If
252         Next i
253         ' check to see if the baseline for effects at MY exists
254         If IsNothing(BaselineEff) Then
255             MsgBox("Cannot find a baseline scenario for effects at model " &
256                    "year: " & MY & ".", MsgBoxStyle.Exclamation, "Cannot find baseline")
257             Exit Sub
258         End If
259
260         ' get the values for the current effect
261         ThisEff = CAFE.EffectsStack(Me.Scenario).Peek()
262
263         For i = 0 To Me.ecLen - 1
264             Me.EC_LifetimeFuelConsumption(i) = Me.Math(ThisEff.EC_LifetimeFuelConsumption(i),
        BaselineEff.EC_LifetimeFuelConsumption(i), OP)
265             Me.EC_Sales(i) = Me.Math(ThisEff.EC_Sales(i), BaselineEff.EC_Sales(i), OP)
266             Me.EC_LifetimeVMT(i) = Me.Math(ThisEff.EC_LifetimeVMT(i), BaselineEff.
        EC_LifetimeVMT(i), OP)
267             Me.EC_AverageFuelEconomy(i) = Me.Math(ThisEff.EC_AverageFuelEconomy(i),
        BaselineEff.EC_AverageFuelEconomy(i), OP)
268         Next i
269
270         For i = 0 To Me.emLen - 1
```

```
271             Me.Em_CO2(i) = Me.Math(ThisEff.Em_CO2(i), BaselineEff.Em_CO2(i), OP)
272             Me.Em_CO(i) = Me.Math(ThisEff.Em_CO(i), BaselineEff.Em_CO(i), OP)
273             Me.Em_VOC(i) = Me.Math(ThisEff.Em_VOC(i), BaselineEff.Em_VOC(i), OP)
274             Me.Em_NOX(i) = Me.Math(ThisEff.Em_NOX(i), BaselineEff.Em_NOX(i), OP)
275             Me.Em_PM(i) = Me.Math(ThisEff.Em_PM(i), BaselineEff.Em_PM(i), OP)
276             Me.Em_SOX(i) = Me.Math(ThisEff.Em_SOX(i), BaselineEff.Em_SOX(i), OP)
277             Me.Em_Sales(i) = Me.Math(ThisEff.Em_Sales(i), BaselineEff.Em_Sales(i), OP)
278             Me.Em_LifetimeVMT(i) = Me.Math(ThisEff.Em_LifetimeVMT(i), BaselineEff.
    Em_LifetimeVMT(i), OP)
279             Me.Em_AverageFuelEconomy(i) = Me.Math(ThisEff.Em_AverageFuelEconomy(i),
    BaselineEff.Em_AverageFuelEconomy(i), OP)
280         Next i
281
282         If CAFE.SafetyIncluded Then
283             For i = 0 To Me.sftLen - 1
284                 'If i <> Me.sftLen - 1 Then
285                 '    Me.Sft_TotalLifetimeFatalities(i) = Double.Epsilon  ' TODO: Report only
    the overall total of "Total Lifetime Fatalities" for now.
286                 'Else
287                 '    Me.Sft_TotalLifetimeFatalities(i) = Me.Math(ThisEff.
    Sft_TotalLifetimeFatalities(i), BaselineEff.Sft_TotalLifetimeFatalities(i), OP)
288                 'End If
289                 Me.Sft_TotalLifetimeFatalities(i) = Me.Math(ThisEff.
    Sft_TotalLifetimeFatalities(i), BaselineEff.Sft_TotalLifetimeFatalities(i), OP)
290                 Me.Sft_Sales(i) = Me.Math(ThisEff.Sft_Sales(i), BaselineEff.Sft_Sales(i), OP)
291                 'Me.Sft_LifetimeVMT(i) = Me.Math(ThisEff.Sft_LifetimeVMT(i), BaselineEff.
    Sft_LifetimeVMT(i), OP)
292                 Me.Sft_AverageCurbWeight(i) = Me.Math(ThisEff.Sft_AverageCurbWeight(i),
    BaselineEff.Sft_AverageCurbWeight(i), OP)
293             Next i
294         End If
295
296         Me.TotalLifetimePretaxFuelExpenditures = Me.Math(ThisEff.
    TotalLifetimePretaxFuelExpenditures, BaselineEff.TotalLifetimePretaxFuelExpenditures, OP)
297         Me.FuelTaxRevenues = Me.Math(ThisEff.FuelTaxRevenues, BaselineEff.FuelTaxRevenues,
    OP)
298         Me.TravelValue = Me.Math(ThisEff.TravelValue, BaselineEff.TravelValue, OP)
299         Me.RefuelingTimeValue = Me.Math(ThisEff.RefuelingTimeValue, BaselineEff.
    RefuelingTimeValue, OP)
300         Me.PetroleumMarketExternalities = Me.Math(ThisEff.PetroleumMarketExternalities,
    BaselineEff.PetroleumMarketExternalities, OP)
301         Me.CongestionCosts = Me.Math(ThisEff.CongestionCosts, BaselineEff.CongestionCosts,
    OP)
302         Me.NoiseCosts = Me.Math(ThisEff.NoiseCosts, BaselineEff.NoiseCosts, OP)
303         Me.AccidentCosts = Me.Math(ThisEff.AccidentCosts, BaselineEff.AccidentCosts, OP)
304         Me.CO2 = Me.Math(ThisEff.CO2, BaselineEff.CO2, OP)
305         Me.CO = Me.Math(ThisEff.CO, BaselineEff.CO, OP)
306         Me.VOC = Me.Math(ThisEff.VOC, BaselineEff.VOC, OP)
307         Me.NOX = Me.Math(ThisEff.NOX, BaselineEff.NOX, OP)
308         Me.PM = Me.Math(ThisEff.PM, BaselineEff.PM, OP)
309         Me.SOX = Me.Math(ThisEff.SOX, BaselineEff.SOX, OP)
310         Me.MonetizedFatalities = Double.Epsilon     ' TODO: monetized fatalities are ignored
    for now
311         'Me.MonetizedFatalities = Me.Math(ThisEff.MonetizedFatalities, BaselineEff.
    MonetizedFatalities, OP)
312
313         Me.DiscTotalLifetimePretaxFuelExpenditures = Me.Math(ThisEff.
    DiscTotalLifetimePretaxFuelExpenditures, BaselineEff.
    DiscTotalLifetimePretaxFuelExpenditures, OP)
314         Me.DiscFuelTaxRevenues = Me.Math(ThisEff.DiscFuelTaxRevenues, BaselineEff.
    DiscFuelTaxRevenues, OP)
315         Me.DiscTravelValue = Me.Math(ThisEff.DiscTravelValue, BaselineEff.DiscTravelValue,
    OP)
316         Me.DiscRefuelingTimeValue = Me.Math(ThisEff.DiscRefuelingTimeValue, BaselineEff.
    DiscRefuelingTimeValue, OP)
317         Me.DiscPetroleumMarketExternalities = Me.Math(ThisEff.
    DiscPetroleumMarketExternalities, BaselineEff.DiscPetroleumMarketExternalities, OP)
318         Me.DiscCongestionCosts = Me.Math(ThisEff.DiscCongestionCosts, BaselineEff.
    DiscCongestionCosts, OP)
```

```
319         Me.DiscNoiseCosts = Me.Math(ThisEff.DiscNoiseCosts, BaselineEff.DiscNoiseCosts, OP)
320         Me.DiscAccidentCosts = Me.Math(ThisEff.DiscAccidentCosts, BaselineEff.
        DiscAccidentCosts, OP)
321         Me.DiscCO2 = Me.Math(ThisEff.DiscCO2, BaselineEff.DiscCO2, OP)
322         Me.DiscCO = Me.Math(ThisEff.DiscCO, BaselineEff.DiscCO, OP)
323         Me.DiscVOC = Me.Math(ThisEff.DiscVOC, BaselineEff.DiscVOC, OP)
324         Me.DiscNOX = Me.Math(ThisEff.DiscNOX, BaselineEff.DiscNOX, OP)
325         Me.DiscPM = Me.Math(ThisEff.DiscPM, BaselineEff.DiscPM, OP)
326         Me.DiscSOX = Me.Math(ThisEff.DiscSOX, BaselineEff.DiscSOX, OP)
327         Me.DiscMonetizedFatalities = Double.Epsilon     ' TODO: monetized fatalities are
        ignored for now
328         'Me.DiscMonetizedFatalities = Me.Math(ThisEff.DiscMonetizedFatalities, BaselineEff.
        DiscMonetizedFatalities, OP)
329     End Sub
330
331     Public Shadows Sub Copy(ByRef dst As EffectStackClass)
332         Me.Copy(dst, Me)
333     End Sub
334
335     Public Shared Shadows Sub Copy(ByRef dst As EffectStackClass, ByRef src As
        EffectStackClass)
336         Dim i As Integer
337
338         dst = New EffectStackClass
339
340         dst.MY = src.MY
341         dst.Scenario = src.Scenario
342
343         ' Energy Consumption (Gas, Diesel, Total; all arrs are of len 3)
344         For i = 0 To ecLen - 1
345             dst.EC_LifetimeFuelConsumption(i) = src.EC_LifetimeFuelConsumption(i)
346             dst.EC_Sales(i) = src.EC_Sales(i)
347             dst.EC_LifetimeVMT(i) = src.EC_LifetimeVMT(i)
348             dst.EC_AverageFuelEconomy(i) = src.EC_AverageFuelEconomy(i)
349         Next i
350
351         ' Emissions (LDDV, ..., HDGV2b, Total; all arrs are of len 9)
352         For i = 0 To emLen - 1
353             dst.Em_CO2(i) = src.Em_CO2(i)
354             dst.Em_CO(i) = src.Em_CO(i)
355             dst.Em_VOC(i) = src.Em_VOC(i)
356             dst.Em_NOX(i) = src.Em_NOX(i)
357             dst.Em_PM(i) = src.Em_PM(i)
358             dst.Em_SOX(i) = src.Em_SOX(i)
359             dst.Em_Sales(i) = src.Em_Sales(i)
360             dst.Em_LifetimeVMT(i) = src.Em_LifetimeVMT(i)
361             dst.Em_AverageFuelEconomy(i) = src.Em_AverageFuelEconomy(i)
362         Next i
363
364
365
366
367
368
369
370
371
372
373
374         ' Undiscounted Owner and Societal Costs
375         dst.TotalLifetimePretaxFuelExpenditures = src.TotalLifetimePretaxFuelExpenditures
376         dst.FuelTaxRevenues = src.FuelTaxRevenues
377         dst.TravelValue = src.TravelValue
378         dst.RefuelingTimeValue = src.RefuelingTimeValue
379         dst.PetroleumMarketExternalities = src.PetroleumMarketExternalities
380         dst.CongestionCosts = src.CongestionCosts
381         dst.NoiseCosts = src.NoiseCosts
382         dst.AccidentCosts = src.AccidentCosts
383         dst.CO2 = src.CO2
```

```
384         dst.CO = src.CO
385         dst.VOC = src.VOC
386         dst.NOX = src.NOX
387         dst.PM = src.PM
388         dst.SOX = src.SOX
389         dst.MonetizedFatalities = src.MonetizedFatalities
390
391         ' Discounted Owner and Societal Costs
392         dst.DiscTotalLifetimePretaxFuelExpenditures = src.
    DiscTotalLifetimePretaxFuelExpenditures
393         dst.DiscFuelTaxRevenues = src.DiscFuelTaxRevenues
394         dst.DiscTravelValue = src.DiscTravelValue
395         dst.DiscRefuelingTimeValue = src.DiscRefuelingTimeValue
396         dst.DiscPetroleumMarketExternalities = src.DiscPetroleumMarketExternalities
397         dst.DiscCongestionCosts = src.DiscCongestionCosts
398         dst.DiscNoiseCosts = src.DiscNoiseCosts
399         dst.DiscAccidentCosts = src.DiscAccidentCosts
400         dst.DiscCO2 = src.DiscCO2
401         dst.DiscCO = src.DiscCO
402         dst.DiscVOC = src.DiscVOC
403         dst.DiscNOX = src.DiscNOX
404         dst.DiscPM = src.DiscPM
405         dst.DiscSOX = src.DiscSOX
406         dst.DiscMonetizedFatalities = src.DiscMonetizedFatalities
407     End Sub
408
409     Public Overloads Overrides Function ToArray() As Object()
410         Dim i, j As Integer
411         Dim size As Integer = 46
412         Dim stack(size - 1) As Object
413
414         ' Energy Consumption (Gas, Diesel, Total; all arrs are of len 3)
415         stack(0) = Me.EC_LifetimeFuelConsumption
416         stack(1) = Me.EC_Sales
417         stack(2) = Me.EC_LifetimeVMT
418         stack(3) = Me.EC_AverageFuelEconomy
419         ' Emissions (LDDV, ..., HDGV2b, Total; all arrs are of len 9)
420         stack(4) = Me.Em_CO2
421         stack(5) = Me.Em_CO
422         stack(6) = Me.Em_VOC
423         stack(7) = Me.Em_NOX
424         stack(8) = Me.Em_PM
425         stack(9) = Me.Em_SOX
426         stack(10) = Me.Em_Sales
427         stack(11) = Me.Em_LifetimeVMT
428         stack(12) = Me.Em_AverageFuelEconomy
429
430
431
432
433
434
435
436         ' Undiscounted Owner and Societal Costs
437         stack(16) = Me.TotalLifetimePretaxFuelExpenditures
438         stack(17) = Me.FuelTaxRevenues
439         stack(18) = Me.TravelValue
440         stack(19) = Me.RefuelingTimeValue
441         stack(20) = Me.PetroleumMarketExternalities
442         stack(21) = Me.CongestionCosts
443         stack(22) = Me.NoiseCosts
444         stack(23) = Me.AccidentCosts
445         stack(24) = Me.CO2
446         stack(25) = Me.CO
447         stack(26) = Me.VOC
448         stack(27) = Me.NOX
449         stack(28) = Me.PM
450         stack(29) = Me.SOX
451         stack(30) = Me.MonetizedFatalities
```

```
        ' Discounted Owner and Societal Costs
        stack(31) = Me.DiscTotalLifetimePretaxFuelExpenditures
        stack(32) = Me.DiscFuelTaxRevenues
        stack(33) = Me.DiscTravelValue
        stack(34) = Me.DiscRefuelingTimeValue
        stack(35) = Me.DiscPetroleumMarketExternalities
        stack(36) = Me.DiscCongestionCosts
        stack(37) = Me.DiscNoiseCosts
        stack(38) = Me.DiscAccidentCosts
        stack(39) = Me.DiscCO2
        stack(40) = Me.DiscCO
        stack(41) = Me.DiscVOC
        stack(42) = Me.DiscNOX
        stack(43) = Me.DiscPM
        stack(44) = Me.DiscSOX
        stack(45) = Me.DiscMonetizedFatalities

        Return stack
    End Function

    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Private Function SumArray(ByVal arr() As Double) As Double
        Dim i As Integer
        Dim sum As Double = 0
        Dim len As Integer = arr.Length

        For i = 0 To len - 1
            sum += arr(i)
        Next i

        Return sum
    End Function

    '''
    ' Converts all NaN, -Inf, and +Inf values to 0.0 and converts Double.Epsilon values to NaN
    '
    Public Sub RemoveNaNs()
        Dim i As Integer

        For i = 0 To Me.ecLen - 1
            Me.NanToZero(Me.EC_LifetimeFuelConsumption(i))
            Me.NanToZero(Me.EC_Sales(i))
            Me.NanToZero(Me.EC_LifetimeVMT(i))
            Me.NanToZero(Me.EC_AverageFuelEconomy(i))
        Next i

        For i = 0 To Me.emLen - 1
            Me.NanToZero(Me.Em_CO2(i))
            Me.NanToZero(Me.Em_CO(i))
            Me.NanToZero(Me.Em_VOC(i))
            Me.NanToZero(Me.Em_NOX(i))
            Me.NanToZero(Me.Em_PM(i))
            Me.NanToZero(Me.Em_SOX(i))
            Me.NanToZero(Me.Em_Sales(i))
            Me.NanToZero(Me.Em_LifetimeVMT(i))
            Me.NanToZero(Me.Em_AverageFuelEconomy(i))
        Next i

        If CAFE.SafetyIncluded Then
            For i = 0 To Me.sftLen - 1
                Me.NanToZero(Me.Sft_TotalLifetimeFatalities(i))
                Me.NanToZero(Me.Sft_Sales(i))
                'Me.NanToZero(Me.Sft_LifetimeVMT(i))
                Me.NanToZero(Me.Sft_AverageCurbWeight(i))
            Next i
        End If

        Me.NanToZero(Me.TotalLifetimePretaxFuelExpenditures)
```

```
520         Me.NanToZero(Me.FuelTaxRevenues)
521         Me.NanToZero(Me.TravelValue)
522         Me.NanToZero(Me.RefuelingTimeValue)
523         Me.NanToZero(Me.PetroleumMarketExternalities)
524         Me.NanToZero(Me.CongestionCosts)
525         Me.NanToZero(Me.NoiseCosts)
526         Me.NanToZero(Me.AccidentCosts)
527         Me.NanToZero(Me.CO2)
528         Me.NanToZero(Me.CO)
529         Me.NanToZero(Me.VOC)
530         Me.NanToZero(Me.NOX)
531         Me.NanToZero(Me.PM)
532         Me.NanToZero(Me.SOX)
533         Me.NanToZero(Me.MonetizedFatalities)
534
535         Me.NanToZero(Me.DiscTotalLifetimePretaxFuelExpenditures)
536         Me.NanToZero(Me.DiscFuelTaxRevenues)
537         Me.NanToZero(Me.DiscTravelValue)
538         Me.NanToZero(Me.DiscRefuelingTimeValue)
539         Me.NanToZero(Me.DiscPetroleumMarketExternalities)
540         Me.NanToZero(Me.DiscCongestionCosts)
541         Me.NanToZero(Me.DiscNoiseCosts)
542         Me.NanToZero(Me.DiscAccidentCosts)
543         Me.NanToZero(Me.DiscCO2)
544         Me.NanToZero(Me.DiscCO)
545         Me.NanToZero(Me.DiscVOC)
546         Me.NanToZero(Me.DiscNOX)
547         Me.NanToZero(Me.DiscPM)
548         Me.NanToZero(Me.DiscSOX)
549         Me.NanToZero(Me.DiscMonetizedFatalities)
550     End Sub
551
552 End Class
553
554 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
555 ' This class stores data per model year
556 '''
557 Public Class EffectsStackClass
558     Inherits CafeStacksClass
559
560     Public Overrides Function Push(ByVal csc As CafeStackClass) As Boolean
561         ' create and copy the new ManufacturerStackClass
562         Dim entry As EffectStackClass
563         EffectStackClass.Copy(entry, csc)
564         '
565         ' add it to the stack
566         Me.MyStack(Me.MySize) = entry
567         Me.MySize += 1
568
569         If Me.MySize >= Me.MyStack.Length Then
570             ' array at max capacity; expand the array
571             ReDim Preserve Me.MyStack(Me.MySize + (Me.CapacityIncrement - 1))
572         End If
573
574         Return True
575     End Function
576
577 End Class
578
```

```
Public Class EnginesClass

#Region "EnginesData"
    Public Valid As Boolean                          ' Run-time generated
    Public ReasonForBeingInvalid As String
    Public MyIndexNumber As Integer                  ' Run-time generated

    Public Shared nMostModels As Integer = 0         ' Stores the number of models for the
    '                                                ' engine with the most models

    Public Engine_Code As Integer
    Public Manufacturer As String
    Public MfgIdx As Integer
    Public Name As String
    Public Origin As String
    Public Fuel As String
    Public EngineOilViscosity As String
    Public Cycle As String
    Public AirFuelRatio As Double
    Public FuelSystem As String
    Public Aspiration As String
    Public ValvetrainDesign As String
    Public ValveActuationTiming As String
    Public ValveLift As String
    Public Cylinders As Integer
    Public Configuration As String
    Public ValvesCylinder As Integer
    Public Deactivation As Double
    Public Displacement As Double
    Public MinCompressionRatio As Double
    Public MaxCompressionRatio As Double
    Public Horsepower_HP As Double
    Public Horsepower_RPM As Double
    Public Torque_HP As Double
    Public Torque_RPM As Double
    Public EngineNotes As String

    Public TechOverLUB As CAFE.TechOverRide
    Public TechOverEFR As CAFE.TechOverRide
    Public TechOverOHC As CAFE.TechOverRide
    Public TechOverVVT As CAFE.TechOverRide
    Public TechOverDISP As CAFE.TechOverRide
    Public TechOverVVLT As CAFE.TechOverRide
    Public TechOverSIDI As CAFE.TechOverRide
    Public TechOverSUP As CAFE.TechOverRide
    Public TechOverCVA As CAFE.TechOverRide
    Public TechOverIVT As CAFE.TechOverRide
    Public TechOverVCR As CAFE.TechOverRide
    Public TechOverDSL As CAFE.TechOverRide

    Public ParentIndex As Integer    ' Index to the parent engine, if there is one

    Public nCompleteModels As Integer        ' number of entries in CompleteModelsList
    Public CompleteModelsList() As Integer   ' Complete list of models which use this engine
    Public nModels As Integer                ' Number of valid models that use this engine
    Public ModelsList() As Integer           ' List of valid models that use this engine
    Public TechCost As Double                ' Technology cost applied to this engine.
    Public TechImprv As Double               ' Technology improvements applied to this
    engine.
    Public TechUsed() As Boolean             ' to keep track of what techs are used on this
    engine
#End Region

    Public Sub New()
        ParentIndex = -1     ' set parent index to -1 to indicate that this
        '                    ' engine does not have a parent engine
        Valid = False
        Me.ReasonForBeingInvalid = "initial setting"
        nModels = 0
        nCompleteModels = 0
```

```
69          TechCost = 0
70          TechImprv = 1
71          ReDim TechUsed(-1)
72      End Sub
73      Public Shared Function Clone(ByVal srcIdx As Integer, ByVal VehList() As Integer) As
        Integer
74          Dim eIdx As Integer = CAFE.Engines.Length
75          Debug.WriteLine("Adding new engine (# " + eIdx.ToString + ")")
76
77          ' If this engine has the same number of models as 'Most', then we'll need to reset
        'Most'
78          Dim RecalcMost As Boolean = (CAFE.Engines(srcIdx).nCompleteModels = nMostModels)
79
80          ReDim Preserve CAFE.Engines(eIdx)
81          CAFE.Engines(eIdx) = New EnginesClass
82          EnginesClass.Copy(CAFE.Engines(eIdx), CAFE.Engines(srcIdx))
83          ' Now remove vehicles being split off from source engine
84          For v As Integer = 0 To VehList.Length - 1
85              Dim vv As Integer = 0
86              Do
87                  If VehList(v) = CAFE.Engines(srcIdx).CompleteModelsList(vv) Then
88                      ' Remove this vehicle
89                      Dim r As Integer
90                      For r = vv + 1 To CAFE.Engines(srcIdx).CompleteModelsList.Length - 1
91                          CAFE.Engines(srcIdx).CompleteModelsList(r - 1) = CAFE.Engines
        (srcIdx).CompleteModelsList(r)
92                      Next
93                      r = CAFE.Engines(srcIdx).CompleteModelsList.Length - 2
94                      ReDim Preserve CAFE.Engines(srcIdx).CompleteModelsList(r)
95                      CAFE.Engines(srcIdx).nCompleteModels = r + 1
96                  Else
97                      vv = vv + 1
98                  End If
99              Loop While vv < CAFE.Engines(srcIdx).CompleteModelsList.Length
100             ' Add the vehicle to the new engine
101             CAFE.Engines(eIdx).AddUser(VehList(v))
102         Next v
103         ' Now populate ModelsList for both
104         CAFE.Engines(srcIdx).CreateModelsList()
105         CAFE.Engines(eIdx).CreateModelsList()
106
107         If RecalcMost Then
108             nMostModels = 0
109             For t As Integer = 0 To CAFE.Engines.Length - 1
110                 nMostModels = Math.Max(nMostModels, CAFE.Engines(t).nCompleteModels)
111             Next
112         End If
113         Return eIdx
114     End Function
115
116     Public Sub Merge(ByVal FromEngineIdx As Integer)
117         ' Merge an engine... assumes that the caller knows that the two engines are
118         ' indentical or not.
119         For Each vidx As Integer In CAFE.Engines(FromEngineIdx).CompleteModelsList
120             Me.AddUser(vidx)
121         Next
122     End Sub
123
124     Public Sub Copy(ByRef Dst As EnginesClass)
125         EnginesClass.Copy(Dst, Me)
126     End Sub
127     Public Shared Sub Copy(ByRef Dst As EnginesClass, ByRef Src As EnginesClass)
128         Dim i As Integer
129
130         Dst.Valid = Src.Valid
131         Dst.MyIndexNumber = Src.MyIndexNumber
132
133         Dst.Engine_Code = Src.Engine_Code
134         Dst.Manufacturer = Src.Manufacturer
135         Dst.Name = Src.Name
```

```
136         Dst.Origin = Src.Origin
137         Dst.Fuel = Src.Fuel
138         Dst.EngineOilViscosity = Src.EngineOilViscosity
139         Dst.Cycle = Src.Cycle
140         Dst.AirFuelRatio = Src.AirFuelRatio
141         Dst.FuelSystem = Src.FuelSystem
142         Dst.Aspiration = Src.Aspiration
143         Dst.ValvetrainDesign = Src.ValvetrainDesign
144         Dst.ValveActuationTiming = Src.ValveActuationTiming
145         Dst.ValveLift = Src.ValveLift
146         Dst.Cylinders = Src.Cylinders
147         Dst.Configuration = Src.Configuration
148         Dst.ValvesCylinder = Src.ValvesCylinder
149         Dst.Deactivation = Src.Deactivation
150         Dst.Displacement = Src.Displacement
151         Dst.MinCompressionRatio = Src.MinCompressionRatio
152         Dst.MaxCompressionRatio = Src.MaxCompressionRatio
153         Dst.Horsepower_HP = Src.Horsepower_HP
154         Dst.Horsepower_RPM = Src.Horsepower_RPM
155         Dst.Torque_HP = Src.Torque_HP
156         Dst.Torque_RPM = Src.Torque_RPM
157         Dst.EngineNotes = Src.EngineNotes
158
159         Dst.TechOverLUB = Src.TechOverLUB
160         Dst.TechOverEFR = Src.TechOverEFR
161         Dst.TechOverOHC = Src.TechOverOHC
162         Dst.TechOverVVT = Src.TechOverVVT
163         Dst.TechOverDISP = Src.TechOverDISP
164         Dst.TechOverVVLT = Src.TechOverVVLT
165         Dst.TechOverSIDI = Src.TechOverSIDI
166         Dst.TechOverSUP = Src.TechOverSUP
167         Dst.TechOverCVA = Src.TechOverCVA
168         Dst.TechOverIVT = Src.TechOverIVT
169         Dst.TechOverVCR = Src.TechOverVCR
170         Dst.TechOverDSL = Src.TechOverDSL
171
172         Dst.nCompleteModels = Src.nCompleteModels
173         ReDim Dst.CompleteModelsList(Dst.nCompleteModels - 1)
174         For i = 0 To Dst.nCompleteModels - 1
175             Dst.CompleteModelsList(i) = Src.CompleteModelsList(i)
176         Next
177
178         Dst.nModels = Src.nModels
179         If Dst.nModels > 0 Then
180             ReDim Dst.ModelsList(Src.nModels - 1)
181             For i = 0 To Src.nModels - 1
182                 Dst.ModelsList(i) = Src.ModelsList(i)
183             Next
184         End If
185         Dst.TechCost = Src.TechCost
186         Dst.TechImprv = Src.TechImprv
187         Dst.MfgIdx = Src.MfgIdx
188         If IsNothing(Src.TechUsed) = False Then
189             ReDim Dst.TechUsed(Src.TechUsed.Length - 1)
190             For i = 0 To Src.TechUsed.Length - 1
191                 Dst.TechUsed(i) = Src.TechUsed(i)
192             Next
193         End If
194     End Sub
195     Public Sub AddUser(ByVal VehIndex As Integer)
196         ' Add a vehicle (index to vehicles) as a vehicle which uses this engine
197         If IsNothing(CompleteModelsList) Then
198             ReDim CompleteModelsList(-1)
199             nCompleteModels = 0
200         End If
201         ReDim Preserve CompleteModelsList(nCompleteModels)
202         CompleteModelsList(nCompleteModels) = VehIndex
203         nCompleteModels = nCompleteModels + 1
204
205         If nCompleteModels > nMostModels Then
```

```
206             nMostModels = nCompleteModels
207         End If
208     End Sub
209
210     '''
211     ' CreateModelsList
212     ' This sub populates the ModelsList from valid vehicles in the CompleteModelsList
213     '
214     Public Sub CreateModelsList()
215         nModels = 0
216         ReDim ModelsList(-1)
217         If Me.Valid Then
218             For v As Integer = 0 To CompleteModelsList.Length - 1
219                 If CAFE.Vehicles(CompleteModelsList(v)).IsValid Then
220                     ' got a valid one
221                     ReDim Preserve ModelsList(ModelsList.Length)
222                     ModelsList(ModelsList.Length - 1) = CompleteModelsList(v)
223                 End If
224             Next
225             nModels = ModelsList.Length
226             Dim NumTechnologies = CAFE.Technologies.GetNumTechnologies
227         End If
228     End Sub
229
230     '''
231     ' Clears the CompleteModelsList() and ModelsList() arrays for the new run.
232     '
233     Public Sub ClearModelLists()
234         ReDim Me.CompleteModelsList(-1)
235         Me.nCompleteModels = 0
236         ReDim Me.ModelsList(-1)
237         Me.nModels = 0
238     End Sub
239
240 #Region "TechOverRide Support Routines"
241     Public Function IsTechOverRideForce(ByVal TechIdx As Integer) As Boolean
242         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Force
243     End Function
244     Public Function IsTechOverRideForce(ByVal TechAbbreviation As String) As Boolean
245         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Force
246     End Function
247     Public Function IsTechOverRideSkip(ByVal TechIdx As Integer) As Boolean
248         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Skip
249     End Function
250     Public Function IsTechOverRideSkip(ByVal TechAbbreviation As String) As Boolean
251         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Skip
252     End Function
253     Public Function IsTechOverRideUsed(ByVal TechIdx As Integer) As Boolean
254         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Used
255     End Function
256     Public Function IsTechOverRideUsed(ByVal TechAbbreviation As String) As Boolean
257         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Used
258     End Function
259     Public Function IsTechOverRideNone(ByVal TechIdx As Integer) As Boolean
260         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.None
261     End Function
262     Public Function IsTechOverRideNone(ByVal TechAbbreviation As String) As Boolean
263         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.None
264     End Function
265     '''
266     ' Get the CAFE.TechOverRIde setting.
267     ' This is rather brute-force with a complete lack of elegance,
268     ' maybe we'll clean this up someday -- but don't count on it
269     ' jvs 15 oct 04
270     Public Function GetTechOverRide(ByVal TechIdx As Integer) As CAFE.TechOverRide
271         Dim Abbr As String
272         ' Get the abbreviation using NRC Vehicle Class 0 (the abbrv are the same for all)
273         Abbr = CAFE.Technologies.VehicleClasses(0).Technology(TechIdx).Abbr
274         Return GetTechOverRide(Abbr)
275     End Function
```

```
276     Public Function GetTechOverRide(ByVal TechAbbreviation As String) As CAFE.TechOverRide
277         Dim RtnCode As CAFE.TechOverRide = CAFE.TechOverRide.None
278         Select Case TechAbbreviation
279             Case "LUB"
280                 RtnCode = Me.TechOverLUB
281             Case "EFR"
282                 RtnCode = Me.TechOverEFR
283             Case "OHC"
284                 RtnCode = Me.TechOverOHC
285             Case "VVT"
286                 RtnCode = Me.TechOverVVT
287             Case "DISP"
288                 RtnCode = Me.TechOverDISP
289             Case "VVLT"
290                 RtnCode = Me.TechOverVVLT
291             Case "SIDI"
292                 RtnCode = Me.TechOverSIDI
293             Case "SUP"
294                 RtnCode = Me.TechOverSUP
295             Case "CVA"
296                 RtnCode = Me.TechOverCVA
297             Case "IVT"
298                 RtnCode = Me.TechOverIVT
299             Case "VCR"
300                 RtnCode = Me.TechOverVCR
301             Case "DSL"
302                 RtnCode = Me.TechOverDSL
303         End Select
304         Return RtnCode
305     End Function
306 #End Region
307
308 End Class
309
```

```
Public Class ManufacturerClass

    ' these 2 vars are for debugging  <ms>
    Public Shared nValidVehiclesForAllMfgs As Integer = 0
    Public Shared nDiscardedVehiclesForAllMfgs As Integer = 0

    Public Shared nValidMfgs As Integer
    Public Shared nInvalidMfgs As Integer

    Public Valid As Boolean
    Public MfgName As String
    Public MfgIndex As Integer      ' stores the index for this MFG

    Public CostAllocationStrategy As Integer    ' determines the cost allocation strategy to use for this mfg
    Public WillingToPayCafeFines() As Boolean   ' determines if this mfg is willing to pay fines

    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Stores the total number of RC's (indexed 0 to 11),
    ' including the unregulated class "0".
    '''
    Public Const nRC As Integer = 12
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Stores the indexes to various vehicles
    ' RC(0)           = stores indexes to unknown vehicles
    ' RC(1)           = stores domestic car indexes
    ' RC(2)           = stores imported car indexes
    ' RC(3) - RC(10) = stores lt. truck indexes for RC's 1-8
    ' RC(11)          = stores indexes for heavy light trucks regulated as a separate class
    '''
    Public RC(nRC - 1)() As Integer

    ' Indexes to the various entities (vehicles, engines, transmissions)
    Public AllVehicleIndexes() As Integer
    Public ValidVehicleIndexes() As Integer
    Public DiscardedVehicleIndexes() As Integer        ' Stores vehicles with bad or missing data
    Public EngineIndexes() As Integer
    Public TransmissionIndexes() As Integer

    ' Nameplates:
    ' NamePlates are based on the "Model" field (and not the Nameplate field) of the
    ' vehicle description file.
    ' Unlike vehicles, engines or transmissions, which are stored at the CAFE level,.
    ' NamePlates are stored at the MFG level.
    ' The NamePlate() array stores the information about the Nameplate,
    ' the NamePlateIndexes() array is there simply to schmooze into the FindTechByGroup
    ' routine in a painless way.
    Public NamePlates() As NamePlateClass           ' stores nameplate info
    Public NamePlateIndexes() As Integer            ' Nameplate Indexes

    Public nAllVehicles As Integer
    Public nValidVehicles As Integer
    Public nDiscardedVehicles As Integer
    Public nEngines As Integer
    Public nTransmissions As Integer
    Public nNamePlates As Integer

    ' TechExcluded represents technologies excluded for this MFG based on info from input files
    Public TechExcluded() As Boolean

    ' Phase-in values:
    ' TechApplied is the count of vehicles with a given technology applied by the model for this MFG
    ' TechExhausted is used to mark when a technology has exceeded it's phase-in threshold. Cleared each MY
    Public TechApplied() As Double
    Public TechExhausted() As Boolean
```

```
65
66
67  #Region " Not used variables; need to remove "
68
69      'Public TruckIndexes() As Integer
70      'Public CarIndexes() As Integer
71      'Public UnknownVehicles() As Integer      ' Vehicle class unknown
72      'Public nTrucks As Integer
73      'Public nCars As Integer
74      'Public nUnknown As Integer
75
76      '#Region " _Orig values "
77      '     '' _Orig values are the CAFE values at start-up for this mfg
78      '     'Public TruckCAFE_Orig As Double
79      '     'Public TruckSTD_Orig As Double
80      '     'Public TruckCredit_Orig As Double
81      '     'Public TruckFines_Orig As Double
82
83      '     'Public CarCAFE_Orig As Double
84      '     'Public CarSTD_Orig As Double
85      '     'Public CarCredit_Orig As Double
86      '     'Public CarFines_Orig As Double
87      '#End Region
88
89      'Public TruckCAFE As Double = Double.NaN      ' The CAFE level for all light trucks
90      'Public TruckSTD As Double = Double.NaN       ' The STD for trucks
91      'Public TruckCredit As Double = Double.NaN    ' The Credit for trucks
92      'Public TruckFines As Double = Double.NaN     ' Fines for trucks
93
94      'Public CarCAFE As Double = Double.NaN        ' CAFE level for all cars
95      'Public CarSTD As Double = Double.NaN         ' STD for cars
96      'Public CarCredit As Double = Double.NaN      ' Credit for cars
97      'Public CarFines As Double = Double.NaN       ' Fines for cars
98
99  #End Region
100
101     ' the _Orig values are the CAFE values at start-up for this mfg, before the model was ↙
        run
102     Public RcTotalSales_Orig(nRC - 1) As Double      ' The initial total sales by ↙
        regulatory class
103     Public RcSalesOverFE_Orig(nRC - 1) As Double     ' The initial total sales/FE by ↙
        regulatory class
104     Public RcCAFE_Orig(nRC - 1) As Double            ' The initial CAFE level by regulatory ↙
         class
105     'Public RcSTD_Orig(nRC - 1) As Double            ' The initial STD by regulatory class
106     Public RcCredit_Orig(nRC - 1) As Double          ' The initial Credit by regulatory ↙
        class
107     Public RcCredit_Last_Converge(nRC - 1) As Double  ' Credits after last convergence ↙
        loop
108     Public RcFines_Orig(nRC - 1) As Double           ' The initial Fines by regulatory ↙
        class
109
110     ' the current modeling values (initial, then final)
111     Public RcTotalSales(nRC - 1) As Double           ' The total sales by regulatory class
112     Public RcSalesOverFE(nRC - 1) As Double          ' The total sales/FE by regulatory ↙
        class
113     Public RcCAFE(nRC - 1) As Double                 ' The CAFE level by regulatory class
114     'Public RcSTD(nRC - 1) As Double                 ' The STD by regulatory class
115     Public RcCredit(nRC - 1) As Double               ' The Credit by regulatory class
116     Public RcFines(nRC - 1) As Double                ' The Fines by regulatory class
117
118     ' the _Temp values are the values generated by the ReCalc methods
119     Public RcTotalSales_Temp(nRC - 1) As Double      ' The temporary total sales by ↙
        regulatory class
120     Public RcSalesOverFE_Temp(nRC - 1) As Double     ' The temporary total sales/FE by ↙
        regulatory class
121     Public RcCAFE_Temp(nRC - 1) As Double            ' The temporary CAFE level by ↙
        regulatory class
122     'Public RcSTD_Temp(nRC - 1) As Double            ' The temporary STD by regulatory ↙
        class
```

```
123     Public RcCredit_Temp(nRC - 1) As Double          ' The temporary Credit by regulatory class
124     Public RcFines_Temp(nRC - 1) As Double           ' The temporary Fines by regulatory class
125
126     ' values used to determine whether to terminate ConvergenceLoop
127     Public ConvergenceLoopRcCAFE(nRC - 1, CAFE.NumberOfAllowedConvergenceLoops - 1) As Double ' CAFE (by RC, ConvergenceLoop)
128     Public ConvergenceLoopRcShare(nRC - 1, CAFE.NumberOfAllowedConvergenceLoops - 1) As Double ' Sales share (by RC, ConvergenceLoop)
129
130     ' overall values
131     'Public TotalTechUsed() As Double                 ' total techs used for all vehicles, by tech
132     'Public TotalTechsUsed As Double                  ' total, over-all, techs used for all vehicles
133     'Private VehAccounted() As Boolean                ' if a veh has multiple techs used, only count its
134     '                                                 ' sales once in TotalTechsUsed
135     ' Trading pool info:
136     Public TradingPool(4)() As Integer
137     '
138 #Region " Variables for Manufacturer reports "
139
140     ' by RC totals
141     Public RcTotalTechCosts(nRC - 1) As Double
142     Public RcTotalPurchaseCosts(nRC - 1) As Double
143     Public RcApplicationRatePerTechnology(nRC - 1)() As Double
144     Public RcPenetrationRatePerTechnology(nRC - 1)() As Double
145     Public RcTotalUSEmploymentHours(nRC - 1) As Double
146
147     ' by RC averages
148     Public RcAverageFuelEconomy(nRC - 1) As Double
149     Public RcAverageCurbWeight(nRC - 1) As Double
150     Public RcAverageArea(nRC - 1) As Double
151     Public RcAverageTechCosts(nRC - 1) As Double
152     Public RcAverageFinesOwed(nRC - 1) As Double
153     Public RcAveragePrice(nRC - 1) As Double
154
155     ' light truck totals
156     Public LtTotalSales As Double
157     Public LtTotalTechCosts As Double
158     Public LtTotalFinesOwed As Double
159     Public LtTotalPurchaseCosts As Double
160     Public LtApplicationRatePerTechnology() As Double
161     Public LtPenetrationRatePerTechnology() As Double
162     Public LtTotalUSEmploymentHours As Double
163
164     ' light truck averages
165     Public LtAverageFuelEconomy As Double
166     Public LtAverageCurbWeight As Double
167     Public LtAverageArea As Double
168     Public LtAverageTechCosts As Double
169     Public LtAverageFinesOwed As Double
170     Public LtAveragePrice As Double
171
172     ' overall totals
173     Public TotalFines_Orig As Double = Double.NaN    ' The initial overall fines
174     Public TotalFines As Double = Double.NaN         ' The overall fines
175     Public TotalFines_Temp As Double = Double.NaN    ' The temporary overall fines
176     Public TotalUSEmploymentHours As Double
177
178     Public TotalSales As Double = Double.NaN         ' the overall sales for all vehicles
179     Public TotalTechCosts As Double                  ' the total cost of all techs applied to all vehicles
180     Public TotalPurchaseCosts As Double
181     Public ApplicationRatePerTechnology() As Double
182     Public PenetrationRatePerTechnology() As Double
183
184     ' overall averages
```

```
185     Public AverageFuelEconomy As Double
186     Public AverageCWBefore As Double                ' the average curb weight before the
        model runs
187     Public AverageCWAfter As Double                 ' the average curb weight after the
        model runs
188     Public AverageAreaBefore As Double              ' the average area before the model
        runs
189     Public AverageAreaAfter As Double               ' the average area after the model
        runs
190     Public AverageTechCosts As Double
191     Public AverageFinesOwed As Double
192     Public AveragePrice As Double
193
194 #End Region
195
196
197 #Region " Variables for Industry Wide reports "
198
199     ' by RC totals
200     Public Shared IW_RcTotalSales(nRC - 1) As Double
201     Public Shared IW_RcTotalTechCosts(nRC - 1) As Double
202     Public Shared IW_RcTotalFinesOwed(nRC - 1) As Double
203     Public Shared IW_RcTotalPurchaseCosts(nRC - 1) As Double
204     Public Shared IW_RcApplicationRatePerTechnology(nRC - 1)() As Double
205     Public Shared IW_RcPenetrationRatePerTechnology(nRC - 1)() As Double
206     Public Shared IW_RcTotalUSEmploymentHours(nRC - 1) As Double
207
208     ' by RC averages
209     Public Shared IW_RcAverageFuelEconomy(nRC - 1) As Double
210     Public Shared IW_RcAverageCurbWeight(nRC - 1) As Double
211     Public Shared IW_RcAverageArea(nRC - 1) As Double
212     Public Shared IW_RcAverageTechCosts(nRC - 1) As Double
213     Public Shared IW_RcAverageFinesOwed(nRC - 1) As Double
214     Public Shared IW_RcAveragePrice(nRC - 1) As Double
215
216     ' light truck totals
217     Public Shared IW_LtTotalSales As Double
218     Public Shared IW_LtTotalTechCosts As Double
219     Public Shared IW_LtTotalFinesOwed As Double
220     Public Shared IW_LtTotalPurchaseCosts As Double
221     Public Shared IW_LtApplicationRatePerTechnology() As Double
222     Public Shared IW_LtPenetrationRatePerTechnology() As Double
223     Public Shared IW_LtTotalUSEmploymentHours As Double
224
225     ' light truck averages
226     Public Shared IW_LtAverageFuelEconomy As Double
227     Public Shared IW_LtAverageCurbWeight As Double
228     Public Shared IW_LtAverageArea As Double
229     Public Shared IW_LtAverageTechCosts As Double
230     Public Shared IW_LtAverageFinesOwed As Double
231     Public Shared IW_LtAveragePrice As Double
232
233     ' overall totals
234     Public Shared IW_TotalSales As Double
235     Public Shared IW_TotalTechCosts As Double
236     Public Shared IW_TotalFinesOwed As Double
237     Public Shared IW_TotalPurchaseCosts As Double
238     Public Shared IW_ApplicationRatePerTechnology() As Double
239     Public Shared IW_PenetrationRatePerTechnology() As Double
240     Public Shared IW_TotalUSEmploymentHours As Double
241
242     ' overall averages
243     Public Shared IW_AverageFuelEconomy As Double
244     Public Shared IW_AverageCurbWeight As Double
245     Public Shared IW_AverageArea As Double
246     Public Shared IW_AverageTechCosts As Double
247     Public Shared IW_AverageFinesOwed As Double
248     Public Shared IW_AveragePrice As Double
249
250 #End Region
```

```
251
252
253     Public Sub New(ByVal ManufacturerName As String)
254         If DBUG Then
255             denter(Me, "New(ManufacturerName)")
256         End If
257
258         Dim i As Integer
259
260         MfgName = ManufacturerName
261
262         Me.TotalTechCosts = 0        ' there are no tech costs initially
263
264         Me.nValidVehicles = 0
265         Me.nDiscardedVehicles = 0
266         Me.nEngines = 0
267         Me.nTransmissions = 0
268         Me.nNamePlates = 0
269
270         ' set all _Orig values to Double.NaN, so that they can be
271         ' identified when calculating cafe, credit, etc. for the first time
272         For i = 0 To nRC - 1
273             Me.RcTotalSales_Orig(i) = Double.NaN
274             Me.RcSalesOverFE_Orig(i) = Double.NaN
275             Me.RcCAFE_Orig(i) = Double.NaN
276             'Me.RcSTD_Orig(i) = Double.NaN
277             Me.RcCredit_Orig(i) = Double.NaN
278             Me.RcFines_Orig(i) = Double.NaN
279         Next i
280     End Sub
281
282
283 #Region " Copy methods "
284
285     Public Sub Copy(ByRef Dst As ManufacturerClass)
286         If DBUG Then
287             denter(Me, "Copy(Dst)")
288         End If
289
290         ManufacturerClass.Copy(Dst, Me)
291     End Sub
292
293     Public Shared Sub Copy(ByRef Dst As ManufacturerClass, ByRef Src As ManufacturerClass)
294         If DBUG Then
295             denter(New ManufacturerClass(""), "Copy(Dst, Src)")
296         End If
297
298         Dim i, j As Integer
299
300         ' first initialize the destination object
301         Dst = New ManufacturerClass(Src.MfgName)
302
303         Dst.Valid = Src.Valid
304         Dst.MfgName = Src.MfgName
305         Dst.MfgIndex = Src.MfgIndex
306
307         For i = 0 To ManufacturerClass.nRC - 1
308             CopyArray(Dst.RC(i), Src.RC(i))
309         Next i
310
311         Dst.CostAllocationStrategy = Src.CostAllocationStrategy
312
313         CopyArray(Dst.WillingToPayCafeFines, Src.WillingToPayCafeFines)
314
315         CopyArray(Dst.AllVehicleIndexes, Src.AllVehicleIndexes)
316         CopyArray(Dst.ValidVehicleIndexes, Src.ValidVehicleIndexes)
317         CopyArray(Dst.DiscardedVehicleIndexes, Src.DiscardedVehicleIndexes)
318         CopyArray(Dst.EngineIndexes, Src.EngineIndexes)
319         CopyArray(Dst.TransmissionIndexes, Src.TransmissionIndexes)
320         CopyArray(Dst.NamePlates, Src.NamePlates)
```

```
321         CopyArray(Dst.NamePlateIndexes, Src.NamePlateIndexes)
322
323         Dst.AllVehicleIndexes = Src.AllVehicleIndexes
324         Dst.nValidVehicles = Src.nValidVehicles
325         Dst.nDiscardedVehicles = Src.nDiscardedVehicles
326         Dst.nEngines = Src.nEngines
327         Dst.nTransmissions = Src.nTransmissions
328         Dst.nNamePlates = Src.nNamePlates
329
330         CopyArray(Dst.RcTotalSales, Src.RcTotalSales)
331         CopyArray(Dst.RcSalesOverFE, Src.RcSalesOverFE)
332         CopyArray(Dst.RcCAFE, Src.RcCAFE)
333         'CopyArray(Dst.RcSTD, Src.RcSTD)
334         CopyArray(Dst.RcCredit, Src.RcCredit)
335         CopyArray(Dst.RcFines, Src.RcFines)
336
337         Dst.TotalFines = Src.TotalFines
338
339         CopyArray(Dst.TechExcluded, Src.TechExcluded)
340         CopyArray(Dst.TechApplied, Src.TechApplied)
341         CopyArray(Dst.TechExhausted, Src.TechExhausted)
342
343     End Sub
344
345     Protected Shared Sub CopyArray(ByRef Dst() As Integer, ByRef Src() As Integer, _
346                                    Optional ByRef SrcLength As Integer = -1)
347         If DBUG Then
348             denter(New ManufacturerClass(""), "CopyArray(Dst(), Src(), [SrcLength = -1])")
349         End If
350
351         Dim i As Integer
352         Dim length As Integer = SrcLength
353
354         If Not IsNothing(Src) Then
355             ' only copy values from source, if there are values to copy
356             '
357             ' get the length
358             If length < 0 Then
359                 length = Src.Length
360             End If
361             ' first dimension the Dst array to be the same size as Src array
362             ReDim Dst(length - 1)
363             ' then copy values
364             For i = 0 To length - 1
365                 Dst(i) = Src(i)
366             Next i
367         Else
368             ' if the array has not been initialized in the source,
369             ' set array to nothing in the destination object
370             Dst = Nothing
371         End If
372     End Sub
373
374     Protected Shared Sub CopyArray(ByRef Dst() As Double, ByRef Src() As Double, _
375                                    Optional ByRef SrcLength As Integer = -1)
376         If DBUG Then
377             denter(New ManufacturerClass(""), "CopyArray(Dst(), Src(), [SrcLength = -1])")
378         End If
379
380         Dim i As Integer
381         Dim length As Integer = SrcLength
382
383         If Not IsNothing(Src) Then
384             ' only copy values from source, if there are values to copy
385             '
386             ' get the length
387             If length < 0 Then
388                 length = Src.Length
389             End If
390             ' first dimension the Dst array to be the same size as Src array
```

```
391             ReDim Dst(length - 1)
392             ' then copy values
393             For i = 0 To length - 1
394                 Dst(i) = Src(i)
395             Next i
396         Else
397             ' if the array has not been initialized in the source,
398             ' set array to nothing in the destination object
399             Dst = Nothing
400         End If
401     End Sub
402 
403     Protected Shared Sub CopyArray(ByRef Dst() As Boolean, ByRef Src() As Boolean, _
404                                    Optional ByRef SrcLength As Integer = -1)
405         If DBUG Then
406             denter(New ManufacturerClass(""), "CopyArray(Dst(), Src(), [SrcLength = -1])")
407         End If
408 
409         Dim i As Integer
410         Dim length As Integer = SrcLength
411 
412         If Not IsNothing(Src) Then
413             ' only copy values from source, if there are values to copy
414             '
415             ' get the length
416             If length < 0 Then
417                 length = Src.Length
418             End If
419             ' first dimension the Dst array to be the same size as Src array
420             ReDim Dst(length - 1)
421             ' then copy values
422             For i = 0 To length - 1
423                 Dst(i) = Src(i)
424             Next i
425         Else
426             ' if the array has not been initialized in the source,
427             ' set array to nothing in the destination object
428             Dst = Nothing
429         End If
430     End Sub
431 
432     Protected Shared Sub CopyArray(ByRef Dst() As NamePlateClass, ByRef Src() As
    NamePlateClass, _
433                                    Optional ByRef SrcLength As Integer = -1)
434         If DBUG Then
435             denter(New ManufacturerClass(""), "CopyArray(Dst(), Src(), [SrcLength = -1])")
436         End If
437 
438         Dim i As Integer
439         Dim length As Integer = SrcLength
440 
441         If Not IsNothing(Src) Then
442             ' only copy values from source, if there are values to copy
443             '
444             ' get the length
445             If length < 0 Then
446                 length = Src.Length
447             End If
448             ' first dimension the Dst array to be the same size as Src array
449             ReDim Dst(length - 1)
450             ' then copy values
451             For i = 0 To length - 1
452                 Dst(i) = New NamePlateClass
453                 NamePlateClass.Copy(Dst(i), Src(i))
454             Next i
455         Else
456             ' if the array has not been initialized in the source,
457             ' set array to nothing in the destination object
458             Dst = Nothing
459         End If
```

```
460     End Sub
461
462 #End Region
463
464
465 #Region " PreProcess() and PostProcess() methods "
466
467     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
468     ' Pre-process method for the Scenario loop.
469     ' Called before the first scenario.
470     '''
471     Public Shared Sub PreProcessForScenarioLoop(ByVal Iteration As Integer)
472         CAFE.Profiling.FunctionStart()
473
474         If DBUG Then
475             denter(Nothing, "PreProcessForScenarioLoop(" + Iteration.ToString + ")")
476         End If
477
478         Dim i As Integer
479         Dim j As Integer
480
481         If Iteration = 0 Then
482             ManufacturerClass.nValidMfgs = 0
483             ManufacturerClass.nInvalidMfgs = 0
484
485             ManufacturerClass.nValidVehiclesForAllMfgs = 0
486             ManufacturerClass.nDiscardedVehiclesForAllMfgs = 0
487
488             ' Initialize for the run
489             ' parse all mfgs
490             ParseMfgs()          ' Flush-out MFG list.
491
492             Dim mfg As ManufacturerClass
493             Dim veh As VehiclesClass
494
495             ' scan each mfg
496             For i = 0 To CAFE.MasterMfgs.Length - 1
497                 Debug.WriteLine("About to validate mfgs(" + i.ToString + ")")
498                 mfg = CAFE.MasterMfgs(i)          ' get current mfg
499
500                 ' first, cross-reference the current mfg
501                 mfg.CrossReference()
502
503                 ' then, split AllVehicleIndexes into valid and invalid
504                 mfg.SplitValidVehicles()
505
506                 ' finally, validate each mfg
507                 If mfg.Validate() Then
508                     ManufacturerClass.nValidMfgs += 1        ' increment valid mfg counter
509                 Else
510                     ManufacturerClass.nInvalidMfgs += 1      ' increment discarded mfg
    counter
511                 End If
512             Next i
513
514             ' Record invalid vehicles
515             Dim DV As WriteDiscardedVehiclesClass = New WriteDiscardedVehiclesClass
516             DV.Close()
517             DV = Nothing
518         End If
519         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
520     End Sub
521
522     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
523     ' Pre-process method for the Model Year loop.
524     ' Called before starting the MY loop.
525     '
526     ' This method copies the CAFE.Master* objects to
527     ' "working" objects and initializes TechApplies.
528     '''
```

```
529     Public Shared Sub PreProcessForMYLoop(ByVal Scenario As Integer)
530         CAFE.Profiling.FunctionStart()
531
532         If DBUG Then
533             denter("ManufacturerClass_Shared", "PreProcessForScenarioLoop()")
534         End If
535
536         Dim i As Integer
537         Dim j As Integer
538         Dim nMfg As Integer
539         Dim nVeh As Integer
540         Dim nEng As Integer
541         Dim nTran As Integer
542         Dim nMY As Integer
543
544         ' initialize data
545         nMfg = CAFE.MasterMfgs.Length
546         nVeh = CAFE.MasterVehicles.Length
547         nEng = CAFE.MasterEngines.Length
548         nTran = CAFE.MasterTransmissions.Length
549         nMY = CAFE.ModelYearsSelected.Length
550
551         ' redim the stacks since we are adding a new scenario
552         ReDim Preserve CAFE.VehiclesStack(Scenario)
553         ReDim Preserve CAFE.MfgsStack(Scenario)
554         ReDim Preserve CAFE.IndustriesStack(Scenario)
555         ReDim Preserve CAFE.EffectsStack(Scenario)
556         ReDim Preserve CAFE.EAsStack(Scenario)
557
558         ' Initialize for the scenario
559         '
560         ' clear mfgs from previous scenario
561         If IsNothing(CAFE.Mfgs) = False Then
562             For i = 0 To CAFE.Mfgs.Length - 1
563                 CAFE.Mfgs(i) = Nothing
564             Next
565             CAFE.Mfgs = Nothing
566         End If
567         ' parse all mfg
568         ParseParameters(Scenario)
569
570         ' resize CAFE.Mfgs()
571         ReDim CAFE.Mfgs(ManufacturerClass.nValidMfgs - 1)
572         j = 0
573
574         ' copy MasterMfgs to Mfgs
575         For i = 0 To nMfg - 1
576             If CAFE.MasterMfgs(i).Valid Then
577                 ' only copy if MasterMfgs(i) is valid
578                 ManufacturerClass.Copy(CAFE.Mfgs(j), CAFE.MasterMfgs(i))
579                 ' Since an invalid MFG won't get copied, the mfgindex field in all the mfgs must be cleaned.
580                 CAFE.Mfgs(j).MfgIndex = j
581                 CAFE.Mfgs(j).RC_Clear()
582
583                 ' Allocate space for the TechApplied array
584                 ReDim CAFE.Mfgs(j).TechApplied(TechnologiesClass.nTotalTechnologies - 1)
585
586                 j += 1
587             End If
588         Next i
589
590         ' set up the vehicles, engines, and transmissions
591         nVeh = CAFE.MasterVehicles.Length
592         nEng = CAFE.MasterEngines.Length
593         nTran = CAFE.MasterTransmissions.Length
594         ReDim CAFE.Vehicles(nVeh - 1)
595         ReDim CAFE.Engines(nEng - 1)
596         ReDim CAFE.Transmissions(nTran - 1)
597
```

```
598         ' Clean out previous copies of Vehicles, transmissions and engines
599         For i = 0 To CAFE.Vehicles.Length - 1
600             If Not IsNothing(CAFE.Vehicles(i)) Then
601                 CAFE.Vehicles = Nothing
602             End If
603         Next i
604         For i = 0 To CAFE.Engines.Length - 1
605             If Not IsNothing(CAFE.Engines(i)) Then
606                 CAFE.Engines(i) = Nothing
607             End If
608         Next i
609         For i = 0 To CAFE.Transmissions.Length - 1
610             If Not IsNothing(CAFE.Transmissions(i)) Then
611                 CAFE.Transmissions(i) = Nothing
612             End If
613         Next i
614         ' copy "Master" objects to "working" objects
615         For i = 0 To nVeh - 1
616             ' For vehicles, set theNRC info first
617             CAFE.MasterVehicles(i).SetNRCClass()
618             CAFE.Vehicles(i) = New VehiclesClass
619             VehiclesClass.Copy(CAFE.Vehicles(i), CAFE.MasterVehicles(i))
620             CAFE.Vehicles(i).MfgIdx = CAFE.GetMfgIndex(CAFE.Vehicles(i).Manufacturer)
621             CAFE.Vehicles(i).ValidateForScenario(Scenario)
622         Next i
623         For i = 0 To nEng - 1
624             CAFE.Engines(i) = New EnginesClass
625             EnginesClass.Copy(CAFE.Engines(i), CAFE.MasterEngines(i))
626             CAFE.Engines(i).MfgIdx = CAFE.GetMfgIndex(CAFE.Engines(i).Manufacturer)
627         Next i
628         For i = 0 To nTran - 1
629             CAFE.Transmissions(i) = New TransmissionsClass
630             TransmissionsClass.Copy(CAFE.Transmissions(i), CAFE.MasterTransmissions(i))
631             CAFE.Transmissions(i).MfgIdx = CAFE.GetMfgIndex(CAFE.Transmissions(i).
    Manufacturer)
632         Next i
633
634         ' For valid vehicles, set Mobile6 class (KG:  reinserted 11/23/04)
635         For i = 0 To nVeh - 1
636             If CAFE.Vehicles(i).ValidForScenario Then
637                 CAFE.Vehicles(i).MOBILE6Class = CAFE.Vehicles(i).AssignMobile6Class()
638             End If
639         Next i
640
641         ' initialize tech applies
642         For i = 0 To CAFE.Vehicles.Length - 1
643             CAFE.Technologies.InitTechnologyApplies(i)
644         Next i
645
646         ' Init techused structures for Engines, Transmissions and NamePlates
647         For i = 0 To CAFE.Engines.Length - 1
648             ReDim CAFE.Engines(i).TechUsed(TechnologiesClass.nTotalTechnologies - 1)
649         Next
650         For i = 0 To CAFE.Transmissions.Length - 1
651             ReDim CAFE.Transmissions(i).TechUsed(TechnologiesClass.nTotalTechnologies - 1)
652         Next
653         For i = 0 To CAFE.Mfgs.Length - 1
654             For j = 0 To CAFE.Mfgs(i).NamePlates.Length - 1
655                 ReDim CAFE.Mfgs(i).NamePlates(j).TechUsed(TechnologiesClass.
    nTotalTechnologies - 1)
656             Next j
657         Next i
658
659         If Scenario = 0 Then
660             ' initialize the baseline aggregate data
661             ReDim CAFE.BaselineAggregateData(nMY - 1)
662         End If
663
664         ' clear out all previous aggregate values for this scenario, since we
665         ' want the effects modeling to be independent between multiple scenarios
```

```
666         ReDim CAFE.ScenarioAggregateData(nMY - 1)
667
668         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
669     End Sub
670
671     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
672     ' Pre-process method for the Convergence loop.
673     ' Called once per Model Year.
674     '
675     ' This method parses all mfgs for the current model year.
676     ' Afterwards, this method cross-references each mfg and
677     ' sets the NRC_Class for all vehicles.  Finally, this
678     ' method sets TechEnabled.
679     '''
680     Public Shared Sub PreProcessForConvergenceLoop(ByVal Scenario As Integer, ByVal MY As ↙
    Integer)
681         CAFE.Profiling.FunctionStart()
682
683         If DBUG Then
684             denter("ManufacturerClass_Shared", "PreProcessForMYLoop()")
685         End If
686
687         Dim i, j, k As Integer
688         Dim nMasterMfg As Integer
689         Dim nTechnologies As Integer
690         Dim vIdx As Integer
691         Dim mfg As ManufacturerClass
692         Dim veh As VehiclesClass
693         Dim techStack As TechnologyStackClass
694
695         ' get the number of master mfgs
696         nMasterMfg = CAFE.MasterMfgs.Length
697
698         ' get the total number of technologies
699         nTechnologies = TechnologiesClass.nTotalTechnologies
700
701         ' initialize "nMostModels" counters to 0 for engines and transmissions
702         EnginesClass.nMostModels = 0
703         TransmissionsClass.nMostModels = 0
704
705         ' reset all shared variables used for industry-wide reporting to 0
706         ResetIndustryWideVariables()
707
708         ' scan each mfg
709         For i = 0 To CAFE.Mfgs.Length - 1
710             mfg = CAFE.Mfgs(i)          ' get current mfg
711
712             ' clear all vehicle indexes from the RC()() array to avoid over-counting.
713             mfg.RC_Clear()
714
715             ' scan each vehicle within this mfg
716             For j = 0 To mfg.nValidVehicles - 1
717                 vIdx = mfg.ValidVehicleIndexes(j)          ' get the index of current ↙
    vehicle
718                 veh = CAFE.Vehicles(vIdx)                  ' get the vehicle at current ↙
    index
719
720                 ' Set the vehicle's "valid" setting for this MY
721                 veh.ValidateForMY(Scenario, MY)
722
723                 If veh.IsValid Then
724                     ' pre-process this vehicle before continuing
725                     veh.PreProcess(Scenario, MY)
726
727                     ' add this vehicle to the by RC vehicle list for this mfg
728                     mfg.RC_Add(veh.RegClass, vIdx)
729                 Else
730                     ' If this veh is invalid for this MY (but IS valid overall and for the ↙
     scenario),
731                     ' then log it to error file
```

```
732                     If veh.ValidOverAll = True And veh.ValidForScenario = True Then
733                         'ModellingLogic.ModelErrorsFile.WriteError("Invalid MY: " + i.
    ToString + " " + veh.ReasonForBeingInvalid)
734                     End If
735                 End If
736             Next j
737
738             ' Populate ModelsList for Engines, Transmissions and Nameplates.  The
    modelslist Is populated with
739             ' valid vehicles from the ComplateModelsList for each engine, tran or
    nameplate
740             For j = 0 To mfg.EngineIndexes.Length - 1
741                 ' Reset the engine user lists
742                 Dim eIdx As Integer = mfg.EngineIndexes(j)
743                 CAFE.Engines(eIdx).CreateModelsList()
744             Next j
745             For j = 0 To mfg.TransmissionIndexes.Length - 1
746                 ' reset the transmission user lists
747                 Dim tIdx As Integer = mfg.TransmissionIndexes(j)
748                 CAFE.Transmissions(tIdx).CreateModelsList()
749             Next j
750             For j = 0 To mfg.NamePlateIndexes.Length - 1
751                 Dim nIdx As Integer = mfg.NamePlateIndexes(j)
752                 mfg.NamePlates(nIdx).CreateModelsList()
753             Next
754             mfg.RC_Sort(-1)          ' Sort all the RC veh index lists
755
756             ' Allocate space (also reinitialize) for TechExhausted array
757             ReDim mfg.TechExhausted(nTechnologies - 1)
758         Next i
759
760         ' initialize tech enabled
761         For i = 0 To CAFE.Vehicles.Length - 1
762             CAFE.Technologies.SetTechnologyEnable(MY, i)
763             CAFE.Vehicles(i).PredecessorsAndLegacies(Scenario, MY)
764         Next i
765
766         For i = 0 To CAFE.Vehicles.Length - 1
767             veh = CAFE.Vehicles(i)          ' get the vehicle
768
769             ' get the initial data to push onto the stack
770             ' these are the inital technologies with which each vehicle starts
771             techStack = New TechnologyStackClass(MY, veh)
772
773             ' push the data onto the stack
774             veh.TechUsedStack.Push(techStack)
775         Next i
776
777         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
778     End Sub
779
780     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
781     ' PreProcess method for the manufacturer loop.
782     ' Called before the Mfg loop
783     ' Initialize TotalSales.
784     '''
785     Public Shared Sub PreProcessForMFGLoop(ByVal Scenario As Integer, ByVal MY As Integer,
     ByVal IConvergence As Integer)
786         CAFE.Profiling.FunctionStart()
787
788         For iMfg As Integer = 0 To CAFE.Mfgs.Length - 1
789             Dim mfg As ManufacturerClass = CAFE.Mfgs(iMfg)
790             Dim tSales As Double
791             tSales = 0
792             For v As Integer = 0 To mfg.nValidVehicles - 1
793                 Dim vIdx As Integer = mfg.ValidVehicleIndexes(v)
794                 If CAFE.Vehicles(vIdx).IsValid Then
795                     tSales += CAFE.Vehicles(vIdx).Sales(CAFE.GMYI(MY))
796                 End If
797             Next v
```

```
798             mfg.TotalSales = tSales
799
800             ' Set up rccredits_last_converge (if converge > 0)
801             If IConvergence > 0 Then
802                 For rc As Integer = 0 To nRC - 1
803                     mfg.RcCredit_Last_Converge(rc) = mfg.RcCredit(rc)
804                 Next
805             End If
806         Next iMfg
807
808         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
809     End Sub
810
811
812     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
813     ' Init method for the manufacturer loop.
814     ' Called within the MFG loop to initialize states for the given MFG.
815     '
816     ' This method calculates the initial cafe, credit, and
817     ' fines for the current mfg for the given MY.  This
818     ' method also calculates the initial average curb weight
819     ' and area.
820     '''
821     Public Sub InitForMFGLoop(ByVal Scenario As Integer, _
822                               ByVal MY As Integer, _
823                               ByVal iConvergence As Integer)
824         '
825         CAFE.Profiling.FunctionStart()
826
827         '
828         If DBUG Then
829             denter(Me, "PreProcessForMFGLoop()")
830         End If
831
832         Dim i As Integer
833         Dim vIdx As Integer
834         Dim veh As VehiclesClass
835
836         '' calculate initial cafe, credit, and fines for this mfg
837         'Me.CalcCAFE(MY)
838         'Me.CalcCredit(Scenario, MY)
839         'Me.CalcFines(Scenario, MY)
840
841         ' calculate average curb weight and area
842         For i = 0 To Me.nValidVehicles - 1
843             vIdx = Me.ValidVehicleIndexes(i)
844             veh = CAFE.Vehicles(vIdx)
845
846             If veh.IsValid Then
847                 ' sum up curb weight
848                 Me.AverageCWBefore += veh.CurbWeight
849                 ' sum up area
850                 Me.AverageAreaBefore += veh.OverAllArea
851
852                 For t As Integer = 0 To TechnologiesClass.nTotalTechnologies - 1
853                     veh.TechIgnored(t) = False
854                 Next t
855             End If
856         Next i
857         '
858         ' divide the sums of cw and area by the number
859         ' of vehicles to get the averages
860         Me.AverageCWBefore /= Me.nValidVehicles
861         Me.AverageAreaBefore /= Me.nValidVehicles
862
863         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
864     End Sub
865
866     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
867     ' General post-process method.
```

```
868     '
869     ' The following variables need to be calculated during
870     ' post-processing, in order to be output correctly in the
871     ' report:
872     ' TotalTechUsed() - total techs used for all vehicles,
873     '                   sorted by tech
874     ' TotalTechsUsed  - total, over-all, techs used for all
875     '                   vehicles
876     ' AverageCWBefore - the average curb weight before the
877     '                   model runs
878     ' AverageCWAfter  - the average curb weight after the
879     '                   model runs
880     '
881     '''
882     Public Sub PostProcess()
883         If DBUG Then
884             denter(Me, "PostProcess()")
885         End If
886
887         ' TODO: ManufacturerClass.PostProcess()
888     End Sub
889
890     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
891     ' Post-process method for the Model Year loop.
892     ' Called after each MY.
893     '''
894     Public Shared Sub PostProcessForMYLoop(ByVal Scenario As Integer)
895         CAFE.Profiling.FunctionStart()
896
897         If DBUG Then
898             denter(New ManufacturerClass(""), "PostProcessForScenarioLoop()")
899         End If
900
901         Dim i, j As Integer
902         Dim nIndStack As Integer
903         Dim MY As Integer
904         Dim MYIdx As Integer
905         Dim Industry As IndustryWideStackClass
906
907         ' check to see if we want to run the effects model
908         ' run the effects model only if the 'RunEffectsModel' option is set to true
909         If CAFE.OtherSettings.RunEffectsModel Then
910             ' run the effects model
911             Dim CafeEffects As New CafeEffectsClass
912             CafeEffects.crunchData(Scenario)
913         End If
914
915         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
916     End Sub
917
918     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
919     ' Post-process method for the Convergence loop.
920     '
921     ' This method checks all _Orig and "regular" objects for
922     ' NaN's and converts them to 0's, if found for all mfgs.
923     '''
924     Public Shared Sub PostProcessForConvergenceLoop(ByVal Scenario As Integer, ByVal MY As ↙
         Integer)
925         CAFE.Profiling.FunctionStart()
926
927         If DBUG Then
928             denter(New ManufacturerClass(""), "PostProcessForMYLoop()")
929         End If
930
931         Dim i, j As Integer
932         Dim nMfg As Integer
933         Dim nVeh As Integer
934         Dim mfg As ManufacturerClass
935         Dim veh As VehiclesClass
936         Dim eng As EnginesClass
```

```
937         Dim trn As TransmissionsClass
938         Dim techStack As TechnologyStackClass
939         Dim vehStack As VehicleStackClass
940         Dim mfgStack As ManufacturerStackClass
941         Dim industryStack As IndustryWideStackClass
942         Dim AVC As AggregateVehiclesClass       ' used for aggregating sales and fe data
943         Dim MYIdx As Integer
944
945         nMfg = CAFE.Mfgs.Length
946         nVeh = CAFE.Vehicles.Length
947         MYIdx = CAFE.GMYI(MY)
948
949         For i = 0 To nMfg - 1
950             mfg = CAFE.Mfgs(i)
951             ' remove NaN's from _Orig and regular objects,
952             ' and replace them with 0's for the current mfg
953             mfg.RemoveNaNs()
954
955             ' compute penetration rate/technology
956             mfg.ComputeApplicationAndPenetrationRatesPerTechnology(MY)
957         Next i
958
959         ' some of industry wide totals and averages were computed during
960         ' ComputeTotalsAndAverages() and ComputeApplicationRatePerTechnology()
961         ' methods; next finish computing the rest of industry wide vars
962         ComputeIndustryWideTotalsAndAverages()
963
964         ''''''''''''''''''''''''''' V E H I C L E S '''''''''''''''''''''''''''
965         If Not CAFE.OperatingModes.MonteCarloAnalysis Then
966             ' Don't generate stacks when in Monte Carlo mode
967             ' Skip generating stacks if in MonteCarlo mode (too much memory gets used)
968             If IsNothing(CAFE.VehiclesStack(Scenario)) Then
969                 ' VehiclesStack for this scenario has not been initialized yet, resize it
970                 ReDim CAFE.VehiclesStack(Scenario)(nVeh - 1)
971                 For i = 0 To nVeh - 1
972                     ' initialize the i-th index of the stack
973                     CAFE.VehiclesStack(Scenario)(i) = New VehiclesStackClass
974                 Next i
975             End If
976         End If
977         If Scenario = 0 Then
978             ReDim Preserve CAFE.BaseLineInfo(CAFE.GetMYearIndex(MY))
979             ReDim Preserve CAFE.BaseLineInfo(CAFE.GetMYearIndex(MY)).FinalFuelEconomy(nVeh ↙
        - 1)
980             For i = 0 To nVeh - 1
981                 CAFE.BaseLineInfo(CAFE.GetMYearIndex(MY)).FinalFuelEconomy(i) = CAFE. ↙
      Vehicles(i).FuelEconomy
982             Next i
983         End If
984         ' copy necessary variables to the stack
985         For i = 0 To nVeh - 1
986             veh = CAFE.Vehicles(i)          ' get vehicle
987
988             ' post-process this vehicle
989             veh.PostProcess(Scenario, MY)
990
991             If CAFE.OperatingModes.MonteCarloAnalysis = False Then
992                 ' do tech stack for this vehicle first
993                 '
994                 ' get the data to push onto the stack
995                 techStack = New TechnologyStackClass(MY, veh)
996                 '
997                 ' push the data onto the stack
998                 veh.TechUsedStack.Push(techStack)
999
1000                ' next, do the veh stack itself
1001                '
1002                ' get data to push onto the stack
1003                vehStack = New VehicleStackClass(MY, veh)
1004
```

```
1005                ' push the data onto the stack
1006                CAFE.VehiclesStack(Scenario)(i).Push(vehStack)
1007            End If
1008        Next i
1009
1010        '''''''''''''''''''' M A N U F A C T U R E R S ''''''''''''''''''''
1011        If CAFE.OperatingModes.MonteCarloAnalysis = False Then
1012            ' Don't generate stacks if in MonteCarlo mode
1013            If IsNothing(CAFE.MfgsStack(Scenario)) Then
1014                ' MfgsStack for this scenario has not been initialized yet, resize it
1015                ReDim CAFE.MfgsStack(Scenario)(nMfg - 1)
1016                For i = 0 To nMfg - 1
1017                    ' initialize the i-th index of the stack
1018                    CAFE.MfgsStack(Scenario)(i) = New ManufacturersStackClass
1019                Next i
1020            End If
1021        End If
1022        ' copy necessary variables to the stack and summary
1023        For i = 0 To nMfg - 1
1024            For j = 0 To 2
1025                If CAFE.OperatingModes.MonteCarloAnalysis = False Then
1026                    ' get the data to push onto the stack
1027                    mfgStack = New ManufacturerStackClass(Scenario, MY, i, j)
1028                    ' push the data onto the stack
1029                    CAFE.MfgsStack(Scenario)(i).Push(mfgStack)
1030                End If
1031            Next j
1032
1033            ' Record summary info
1034            CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.SummaryTypeEnum.
    MfgTotalIncurredTechCost, _
1035                                             Scenario, MY, i, CAFE.Mfgs(i).TotalTechCosts /
     1000000)
1036            CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.SummaryTypeEnum.
    MfgFines, _
1037                                             Scenario, MY, i, CAFE.Mfgs(i).TotalFines /
    1000000)
1038            CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.SummaryTypeEnum.
    MfgAveragePriceIncr, _
1039                                             Scenario, MY, i, CAFE.Mfgs(i).AveragePrice)
1040            CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.SummaryTypeEnum.
    MfgAverageTechCost, _
1041                                             Scenario, MY, i, CAFE.Mfgs(i).
    AverageTechCosts)
1042        Next i
1043
1044        '''''''''''''''''''' I N D U S T R Y - W I D E ''''''''''''''''''''
1045        If CAFE.OperatingModes.MonteCarloAnalysis = False Then
1046            ' Don't generate stacks if in MonteCarlo mode
1047            If IsNothing(CAFE.IndustriesStack(Scenario)) Then
1048                ' IndustriesStack for this scenario has not been initialized yet,
    initialize it
1049                CAFE.IndustriesStack(Scenario) = New IndustriesWideStackClass
1050            End If
1051            ' copy necessary variables to the stack
1052            For i = 0 To 2
1053                ' get the data to push onto the stack
1054                industryStack = New IndustryWideStackClass(Scenario, MY, i)
1055                ' push the data onto the stack
1056                CAFE.IndustriesStack(Scenario).Push(industryStack)
1057            Next i
1058        End If
1059        ' Record summary info
1060        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.SummaryTypeEnum.
    TotalIncurredTechCost, _
1061                                         Scenario, MY, -1, ManufacturerClass.
    IW_TotalTechCosts / 1000000)
1062        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.SummaryTypeEnum.
    TotalFines, _
1063                                         Scenario, MY, -1, ManufacturerClass.
```

```
    IW_TotalFinesOwed / 1000000)
1064        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.SummaryTypeEnum.
    OverAllAveragePriceIncr, _
1065                                         Scenario, MY, -1, ManufacturerClass.
    IW_AveragePrice)
1066        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.SummaryTypeEnum.
    OverAllAverageTechCost, _
1067                                         Scenario, MY, -1, ManufacturerClass.
    IW_AverageTechCosts)
1068        CAFE.SummaryOutput.RecordSummary(MyDEBUG.SummaryInfoClass.SummaryTypeEnum.
    TotalSales, _
1069                                         Scenario, MY, -1, ManufacturerClass.IW_TotalSales)
1070
1071        ' aggregate data if we are running the effects model
1072        If CAFE.OtherSettings.RunEffectsModel Then
1073            ' initialize aggregate values based on the scenario
1074            If Scenario = 0 Then
1075                ' if scenario is 0, then we assume that this is the baseline scenario
1076                CAFE.BaselineAggregateData(MYIdx) = New AggregateVehiclesClass(Scenario,
    MY)
1077            End If
1078
1079            ' initialize the alternative scenario
1080            CAFE.ScenarioAggregateData(MYIdx) = New AggregateVehiclesClass(Scenario, MY)
1081        End If
1082
1083        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1084    End Sub
1085
1086    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1087    ' Post-process method for the manufacturer loop.
1088    '''
1089    Public Shared Sub PostProcessForMFGLoop(ByVal Scenario As Integer, _
1090                                            ByVal MY As Integer, _
1091                                            ByVal iConvergence As Integer)
1092        '
1093        CAFE.Profiling.FunctionStart()
1094
1095        '
1096        Dim mfg As ManufacturerClass
1097
1098        For imfg As Integer = 0 To CAFE.Mfgs.Length - 1
1099            mfg = CAFE.Mfgs(imfg)
1100            '    mfg.PostProcessForMFGLoopBeforeCostAllocation()
1101            '    mfg.PostProcessForMFGLoopAfterCostAllocation()
1102            '    mfg.PostProcessForMFGLoopAfterMixShift()
1103
1104            ' Set up rccredits_last_converge (if converge > 0)
1105            For rc As Integer = 0 To nRC - 1
1106                mfg.RcCredit_Last_Converge(rc) = mfg.RcCredit(rc)
1107            Next
1108        Next imfg
1109
1110        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1111    End Sub
1112
1113    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1114    '' Post-process method for the manufacturer loop.
1115    ''''
1116    'Public Shared Sub PostProcessForMFGLoopBeforeCostAllocation()
1117    '    If DBUG Then
1118    '        denter(Me, "PostProcessForMFGLoopBeforeCostAllocation()")
1119    '    End If
1120
1121    'End Sub
1122
1123    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1124    '' Post-process method for the manufacturer loop.
1125    ''''
1126    'Public Shared Sub PostProcessForMFGLoopAfterCostAllocation()
```

```
1127        '    If DBUG Then
1128        '        denter(Me, "PostProcessForMFGLoopAfterCostAllocation()")
1129        '    End If
1130
1131        'End Sub
1132
1133        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1134        '' Post-process method for the manufacturer loop.
1135        ''''
1136        'Public Shared Sub PostProcessForMFGLoopAfterMixShift()
1137        '    If DBUG Then
1138        '        denter(Me, "PostProcessForMFGLoopAfterMixShift()")
1139        '    End If
1140
1141        'End Sub
1142
1143    Parse Mfgs methods
1144
1145        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1146        ' RemoveNaNs method.
1147        '
1148        ' This method checks all _Orig and "regular" objects for
1149        ' NaN's and converts them to 0's, if found.  This method
1150        ' works with this mfg.
1151        '''
1152        Public Sub RemoveNaNs()
1153            Dim i As Integer
1154
1155            ' scan each _Orig and "regular" variable and change all NaN's to 0's
1156            For i = 0 To nRC - 1
1157                If Double.IsNaN(Me.RcTotalSales(i)) Then Me.RcTotalSales(i) = 0
1158                If Double.IsNaN(Me.RcSalesOverFE(i)) Then Me.RcSalesOverFE(i) = 0
1159                If Double.IsNaN(Me.RcCAFE(i)) Then Me.RcCAFE(i) = 0
1160                'If Double.IsNaN(Me.RcSTD(i)) Then Me.RcSTD(i) = 0
1161                If Double.IsNaN(Me.RcCredit(i)) Then Me.RcCredit(i) = 0
1162                If Double.IsNaN(Me.RcFines(i)) Then Me.RcFines(i) = 0
1163
1164                If Double.IsNaN(Me.RcTotalSales_Orig(i)) Then Me.RcTotalSales_Orig(i) = 0
1165                If Double.IsNaN(Me.RcSalesOverFE_Orig(i)) Then Me.RcSalesOverFE_Orig(i) = 0
1166                If Double.IsNaN(Me.RcCAFE_Orig(i)) Then Me.RcCAFE_Orig(i) = 0
1167                'If Double.IsNaN(Me.RcSTD_Orig(i)) Then Me.RcSTD_Orig(i) = 0
1168                If Double.IsNaN(Me.RcCredit_Orig(i)) Then Me.RcCredit_Orig(i) = 0
1169                If Double.IsNaN(Me.RcFines_Orig(i)) Then Me.RcFines_Orig(i) = 0
1170            Next i
1171            '
1172            ' also check total Fines variables
1173            If Double.IsNaN(Me.TotalFines) Then Me.TotalFines = 0
1174            If Double.IsNaN(Me.TotalFines_Orig) Then Me.TotalFines_Orig = 0
1175        End Sub
1176
1177        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1178        ' This method resets all shared variables and array used
1179        ' for industry-wide reporting to 0 and to nothing
1180        '''
1181        Private Shared Sub ResetIndustryWideVariables()
1182            Dim i As Integer
1183
1184            ' by RC
1185            For i = 0 To nRC - 1
1186                ' by RC totals
1187                IW_RcTotalSales(i) = 0
1188                IW_RcTotalTechCosts(i) = 0
1189                IW_RcTotalFinesOwed(i) = 0
1190                IW_RcTotalPurchaseCosts(i) = 0
1191                IW_RcPenetrationRatePerTechnology(i) = Nothing
1192                IW_RcApplicationRatePerTechnology(i) = Nothing
1193                IW_RcTotalUSEmploymentHours(i) = 0
1194                ' by RC averages
1195                IW_RcAverageFuelEconomy(i) = 0
1196                IW_RcAverageCurbWeight(i) = 0
```

```
1197            IW_RcAverageArea(i) = 0
1198            IW_RcAverageTechCosts(i) = 0
1199            IW_RcAverageFinesOwed(i) = 0
1200            IW_RcAveragePrice(i) = 0
1201        Next i
1202        '
1203        ' light truck overall
1204        '
1205        ' lt overall totals
1206        IW_LtTotalSales = 0
1207        IW_LtTotalTechCosts = 0
1208        IW_LtTotalFinesOwed = 0
1209        IW_LtTotalPurchaseCosts = 0
1210        IW_LtPenetrationRatePerTechnology = Nothing
1211        IW_LtApplicationRatePerTechnology = Nothing
1212        IW_LtTotalUSEmploymentHours = 0
1213        ' lt overall averages
1214        IW_LtAverageFuelEconomy = 0
1215        IW_LtAverageCurbWeight = 0
1216        IW_LtAverageArea = 0
1217        IW_LtAverageTechCosts = 0
1218        IW_LtAverageFinesOwed = 0
1219        IW_LtAveragePrice = 0
1220        '
1221        ' overall
1222        '
1223        ' overall totals
1224        IW_TotalSales = 0
1225        IW_TotalTechCosts = 0
1226        IW_TotalFinesOwed = 0
1227        IW_TotalPurchaseCosts = 0
1228        IW_PenetrationRatePerTechnology = Nothing
1229        IW_ApplicationRatePerTechnology = Nothing
1230        IW_TotalUSEmploymentHours = 0
1231        ' overall averages
1232        IW_AverageFuelEconomy = 0
1233        IW_AverageCurbWeight = 0
1234        IW_AverageArea = 0
1235        IW_AverageTechCosts = 0
1236        IW_AverageFinesOwed = 0
1237        IW_AveragePrice = 0
1238    End Sub
1239
1240    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1241    ' This method computes totals and averages:
1242    '   IW_TotalFinesOwed
1243    '   IW_TotalPurchaseCosts
1244    '   IW_ApplicationRatePerTechnology
1245    '   IW_RcTotalFinesOwed
1246    '   IW_RcTotalPurchaseCosts
1247    '   IW_RcApplicationRatePerTechnology
1248    '   IW_AverageFuelEconomy
1249    '   IW_AverageCurbWeight
1250    '   IW_AverageArea
1251    '   IW_AverageTechCosts
1252    '   IW_AverageFinesOwed
1253    '   IW_AveragePrice
1254    '''
1255    Private Shared Sub ComputeIndustryWideTotalsAndAverages()
1256        Dim i, j As Integer
1257        Dim nMfg As Integer
1258        Dim nTech As Integer
1259        Dim mfg As ManufacturerClass
1260
1261        nMfg = CAFE.Mfgs.Length
1262        nTech = TechnologiesClass.nTotalTechnologies
1263
1264        ' first totals
1265        For i = 0 To nMfg - 1
1266            mfg = CAFE.Mfgs(i)
```

```
1267
1268            ' add total fines owed for this mfg
1269            IW_TotalFinesOwed += mfg.TotalFines
1270            ' ..also by RC
1271            For j = 0 To nRC - 1
1272                IW_RcTotalFinesOwed(j) += mfg.RcFines(j)
1273            Next j
1274        Next i
1275
1276        ' then averages
1277        ' divide the sums by the number of vehicles to get the actual averages
1278        IW_AverageFuelEconomy = IW_TotalSales / IW_AverageFuelEconomy
1279        IW_AverageCurbWeight /= IW_TotalSales
1280        IW_AverageArea /= IW_TotalSales
1281        IW_AverageTechCosts = IW_TotalTechCosts / IW_TotalSales
1282        IW_AverageFinesOwed = IW_TotalFinesOwed / IW_TotalSales
1283        IW_AveragePrice = IW_TotalPurchaseCosts / IW_TotalSales
1284        '
1285        For i = 0 To nRC - 1
1286            If (Not IsNothing(IW_RcTotalSales)) AndAlso (Not Double.IsNaN(IW_RcTotalSales ↙
    (i))) _
1287                                                AndAlso IW_RcTotalSales(i) <> 0 Then
1288                '
1289                IW_RcAverageFuelEconomy(i) = IW_RcTotalSales(i) / IW_RcAverageFuelEconomy ↙
    (i)
1290                IW_RcAverageCurbWeight(i) /= IW_RcTotalSales(i)
1291                IW_RcAverageArea(i) /= IW_RcTotalSales(i)
1292                IW_RcAverageTechCosts(i) = IW_RcTotalTechCosts(i) / IW_RcTotalSales(i)
1293                IW_RcAverageFinesOwed(i) = IW_RcTotalFinesOwed(i) / IW_RcTotalSales(i)
1294                IW_RcAveragePrice(i) = IW_RcTotalPurchaseCosts(i) / IW_RcTotalSales(i)
1295            End If
1296        Next i
1297        '
1298        IW_LtAverageFuelEconomy = IW_LtTotalSales / IW_LtAverageFuelEconomy
1299        IW_LtAverageCurbWeight /= IW_LtTotalSales
1300        IW_LtAverageArea /= IW_LtTotalSales
1301        IW_LtAverageTechCosts = IW_LtTotalTechCosts / IW_LtTotalSales
1302        IW_LtAverageFinesOwed = IW_LtTotalFinesOwed / IW_LtTotalSales
1303        IW_LtAveragePrice = IW_LtTotalPurchaseCosts / IW_LtTotalSales
1304
1305        ' then penetration rate
1306        For i = 0 To nTech - 1
1307            ' after all vehicles were scaned, divide ApplicationRatePerTechnology
1308            ' counter for this tech by nVeh to get the fraction/percentage of
1309            ' vehicles that use this technology
1310            IW_ApplicationRatePerTechnology(i) /= IW_TotalSales
1311            IW_PenetrationRatePerTechnology(i) /= IW_TotalSales
1312            '
1313            IW_LtApplicationRatePerTechnology(i) /= IW_LtTotalSales
1314            IW_LtPenetrationRatePerTechnology(i) /= IW_LtTotalSales
1315            '
1316            For j = 0 To nRC - 1
1317                If IW_RcTotalSales(j) <> 0 Then
1318                    IW_RcApplicationRatePerTechnology(j)(i) /= IW_RcTotalSales(j)
1319                    IW_RcPenetrationRatePerTechnology(j)(i) /= IW_RcTotalSales(j)
1320                End If
1321            Next j
1322        Next i
1323    End Sub
1324
1325    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1326    ' This method computes totals and averages:
1327    '   TotalSales
1328    '   TotalTechCosts
1329    '   TotalUSEmploymentHours
1330    '   TotalPurchaseCosts
1331    '   AverageFuelEconomy
1332    '   AverageCWAfter
1333    '   AverageAreaAfter
1334    '   AverageTechCosts
```

```
1335        '   AverageFinesOwed
1336        '   AveragePrice
1337        '   RcTotalTechCosts
1338        '   RcTotalUSEmploymentHours
1339        '   RcTotalPurchaseCosts
1340        '   RcAverageFuelEconomy
1341        '   RcAverageCurbWeight
1342        '   RcAverageArea
1343        '   RcAverageTechCosts
1344        '   RcAverageFinesOwed
1345        '   RcAveragePrice
1346        '   IW_TotalSales
1347        '   IW_TotalTechCosts
1348        '   IW_RcTotalUSEmploymentHours
1349        '   IW_RcTotalSales
1350        '   IW_RcTotalTechCosts
1351        '   IW_RcTotalUSEmploymentHours
1352        '''
1353        Public Sub ComputeTotalsAndAverages(ByVal MY As Integer)
1354            ComputeTotals(MY)
1355            ComputeAverages(MY)
1356        End Sub
1357
1358        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1359        ' This method computes the following:
1360        '   TotalSales
1361        '   TotalTechCosts
1362        '   TotalUSEmploymentHours
1363        '   TotalPurchaseCosts
1364        '   RcTotalTechCosts
1365        '   RcTotalUSEmploymentHours
1366        '   RcTotalPurchaseCosts
1367        '   [Shared] IW_TotalSales
1368        '   [Shared] IW_TotalTechCosts
1369        '   [Shared] IW_TotalPurchaseCosts
1370        '   [Shared] IW_TotalUSEmploymentHours
1371        '   [Shared] IW_RcTotalSales
1372        '   [Shared] IW_RcTotalTechCosts
1373        '   [Shared] IW_RcTotalUSEmploymentHours
1374        '   [Shared] IW_RcTotalPurchaseCosts
1375        '''
1376        Private Sub ComputeTotals(ByVal MY As Integer)
1377            Dim i As Integer
1378            Dim vIdx As Integer
1379            Dim veh As VehiclesClass
1380            Dim vehRC As Integer
1381            Dim vehHours As Double
1382            Dim vehSales As Double
1383
1384            ' reset totals to 0 to avoid over-counting
1385            Me.TotalSales = 0
1386            Me.TotalTechCosts = 0
1387            Me.TotalPurchaseCosts = 0
1388            Me.TotalUSEmploymentHours = 0
1389            '
1390            Me.LtTotalSales = 0
1391            Me.LtTotalTechCosts = 0
1392            Me.LtTotalFinesOwed = 0
1393            Me.LtTotalPurchaseCosts = 0
1394            Me.LtTotalUSEmploymentHours = 0
1395            '
1396            For i = 0 To nRC - 1
1397                Me.RcTotalTechCosts(i) = 0
1398                Me.RcTotalPurchaseCosts(i) = 0
1399                Me.RcTotalUSEmploymentHours(i) = 0
1400            Next i
1401            '
1402            ' compute totals
1403            For i = 0 To Me.nValidVehicles - 1
1404                vIdx = Me.ValidVehicleIndexes(i)
```

```
1405            veh = CAFE.Vehicles(vIdx)
1406            If veh.IsValid Then
1407                vehRC = veh.RegClass
1408                vehHours = veh.EmploymentHoursPerVehicle
1409                vehSales = veh.Sales(CAFE.GMYI(MY))
1410
1411                Me.TotalSales += vehSales                                   ' sum up
    total sales
1412                Me.TotalTechCosts += (veh.TechCost * vehSales)              ' sum up
    total tech costs
1413                Me.TotalPurchaseCosts += (veh.RegCost * vehSales)           ' sum up
                    ' ...chase costs
1414                                                                            
                    ' ...
1415                '
1416                Me.RcTotalTechCosts(vehRC) += (veh.TechCost * vehSales)     ' sum up
    total tech costs by RC
1417                Me.RcTotalPurchaseCosts(vehRC) += (veh.RegCost * vehSales)  ' sum up
    total purchase costs by RC
1418                                                                            ' sum up

1419            End If
1420        Next i
1421        '
1422        ' sum up industry-wide totals
1423        IW_TotalSales += Me.TotalSales
1424        IW_TotalTechCosts += Me.TotalTechCosts
1425        IW_TotalPurchaseCosts += Me.TotalPurchaseCosts
1426        IW_TotalUSEmploymentHours += Me.TotalUSEmploymentHours
1427        '
1428        For i = 0 To nRC - 1
1429            IW_RcTotalSales(i) += Me.RcTotalSales(i)
1430            IW_RcTotalTechCosts(i) += Me.RcTotalTechCosts(i)
1431            IW_RcTotalPurchaseCosts(i) += Me.RcTotalPurchaseCosts(i)
1432            IW_RcTotalUSEmploymentHours(i) += Me.RcTotalUSEmploymentHours(i)
1433            '
1434            If i >= 3 Then
1435                Me.LtTotalSales += Me.RcTotalSales(i)
1436                Me.LtTotalTechCosts += Me.RcTotalTechCosts(i)
1437                Me.LtTotalFinesOwed += Me.RcFines(i)
1438                Me.LtTotalPurchaseCosts += Me.RcTotalPurchaseCosts(i)
1439                Me.LtTotalUSEmploymentHours += Me.RcTotalUSEmploymentHours(i)
1440                '
1441                IW_LtTotalSales += Me.RcTotalSales(i)
1442                IW_LtTotalTechCosts += Me.RcTotalTechCosts(i)
1443                IW_LtTotalFinesOwed += Me.RcFines(i)
1444                IW_LtTotalPurchaseCosts += Me.RcTotalPurchaseCosts(i)
1445                IW_LtTotalUSEmploymentHours += Me.RcTotalUSEmploymentHours(i)
1446            End If
1447        Next i
1448    End Sub
1449
1450    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1451    ' This method computes the following:
1452    '   AverageFuelEconomy
1453    '   AverageCWAfter
1454    '   AverageAreaAfter
1455    '   AverageTechCosts
1456    '   AverageFinesOwed
1457    '   AveragePrice
1458    '   RcAverageFuelEconomy
1459    '   RcAverageCurbWeight
1460    '   RcAverageArea
1461    '   RcAverageTechCosts
1462    '   RcAverageFinesOwed
1463    '   RcAveragePrice
1464    '''
1465    Private Sub ComputeAverages(ByVal MY As Integer)
1466        Dim i As Integer
1467        Dim vIdx As Integer
```

```
1468        Dim veh As VehiclesClass
1469        Dim vehRC As Integer
1470        Dim vehSales As Double
1471
1472        ' reset totals to 0 to avoid over-counting
1473        Me.AverageFuelEconomy = 0
1474        Me.AverageCWAfter = 0
1475        Me.AverageAreaAfter = 0
1476        Me.AverageTechCosts = 0
1477        Me.AverageFinesOwed = 0
1478        Me.AveragePrice = 0
1479        '
1480        Me.LtAverageFuelEconomy = 0
1481        Me.LtAverageCurbWeight = 0
1482        Me.LtAverageArea = 0
1483        Me.LtAverageTechCosts = 0
1484        Me.LtAverageFinesOwed = 0
1485        Me.LtAveragePrice = 0
1486        '
1487        For i = 0 To nRC - 1
1488            Me.RcAverageFuelEconomy(i) = 0
1489            Me.RcAverageCurbWeight(i) = 0
1490            Me.RcAverageArea(i) = 0
1491            Me.RcAverageTechCosts(i) = 0
1492            Me.RcAverageFinesOwed(i) = 0
1493            Me.RcAveragePrice(i) = 0
1494        Next i
1495        '
1496        ' compute totals
1497        For i = 0 To Me.nValidVehicles - 1
1498            vIdx = Me.ValidVehicleIndexes(i)
1499            veh = CAFE.Vehicles(vIdx)
1500            If veh.IsValid Then
1501                vehRC = veh.RegClass
1502                vehSales = veh.Sales(CAFE.GMYI(MY))
1503
1504                ' overall
1505                Me.AverageFuelEconomy += (vehSales / veh.FuelEconomy)           ' sum up ↙
    fuel economy
1506                Me.AverageCWAfter += (veh.CurbWeight * vehSales)                ' sum up ↙
    curb weight
1507                Me.AverageAreaAfter += (veh.OverAllArea * vehSales)             ' sum up ↙
    area
1508                ' by RC
1509                Me.RcAverageFuelEconomy(vehRC) += (vehSales / veh.FuelEconomy)  ' sum up ↙
    fuel economy by RC
1510                Me.RcAverageCurbWeight(vehRC) += (veh.CurbWeight * vehSales)    ' sum up ↙
    curb weight by RC
1511                Me.RcAverageArea(vehRC) += (veh.OverAllArea * vehSales)         ' sum up ↙
    area by RC
1512            End If
1513        Next i
1514        '
1515        ' sum up industry-wide totals
1516        ' dont worry about averaging them out right not, in the                  ↙
    ComputeIndustryWideTotalsAndAverages()
1517        ' sub, these "total" values will be divided by total sales for the entire industry
1518        IW_AverageFuelEconomy += Me.AverageFuelEconomy
1519        IW_AverageCurbWeight += Me.AverageCWAfter
1520        IW_AverageArea += Me.AverageAreaAfter
1521        '
1522        For i = 0 To nRC - 1
1523            IW_RcAverageFuelEconomy(i) += Me.RcAverageFuelEconomy(i)
1524            IW_RcAverageCurbWeight(i) += Me.RcAverageCurbWeight(i)
1525            IW_RcAverageArea(i) += Me.RcAverageArea(i)
1526            '
1527            ' also sum for all light trucks
1528            If i >= 3 Then
1529                Me.LtAverageFuelEconomy += Me.RcAverageFuelEconomy(i)
1530                Me.LtAverageCurbWeight += Me.RcAverageCurbWeight(i)
```

```
1531                Me.LtAverageArea += Me.RcAverageArea(i)
1532                '
1533                IW_LtAverageFuelEconomy += Me.RcAverageFuelEconomy(i)
1534                IW_LtAverageCurbWeight += Me.RcAverageCurbWeight(i)
1535                IW_LtAverageArea += Me.RcAverageArea(i)
1536            End If
1537        Next i
1538        '
1539        ' divide the sums by the number of vehicles to get the actual averages
1540        Me.AverageFuelEconomy = Me.TotalSales / Me.AverageFuelEconomy
1541        Me.AverageCWAfter /= Me.TotalSales
1542        Me.AverageAreaAfter /= Me.TotalSales
1543        Me.AverageTechCosts = Me.TotalTechCosts / Me.TotalSales
1544        Me.AverageFinesOwed = Me.TotalFines / Me.TotalSales
1545        Me.AveragePrice = Me.TotalPurchaseCosts / Me.TotalSales
1546        '
1547        For i = 0 To nRC - 1
1548            If (Not IsNothing(Me.RcTotalSales)) AndAlso (Not Double.IsNaN(Me.RcTotalSales ↙
    (i))) _
1549                                                AndAlso (Me.RcTotalSales(i) <> 0) Then
1550                '
1551                Me.RcAverageFuelEconomy(i) = Me.RcTotalSales(i) / Me.RcAverageFuelEconomy ↙
    (i)
1552                Me.RcAverageCurbWeight(i) /= Me.RcTotalSales(i)
1553                Me.RcAverageArea(i) /= Me.RcTotalSales(i)
1554                Me.RcAverageTechCosts(i) = Me.RcTotalTechCosts(i) / Me.RcTotalSales(i)
1555                Me.RcAverageFinesOwed(i) = Me.RcFines(i) / Me.RcTotalSales(i)
1556                Me.RcAveragePrice(i) = Me.RcTotalPurchaseCosts(i) / Me.RcTotalSales(i)
1557            End If
1558        Next i
1559        '
1560        Me.LtAverageFuelEconomy = Me.LtTotalSales / Me.LtAverageFuelEconomy
1561        Me.LtAverageCurbWeight /= Me.LtTotalSales
1562        Me.LtAverageArea /= Me.LtTotalSales
1563        Me.LtAverageTechCosts = Me.LtTotalTechCosts / Me.LtTotalSales
1564        Me.LtAverageFinesOwed = Me.LtTotalFinesOwed / Me.LtTotalSales
1565        Me.LtAveragePrice = Me.LtTotalPurchaseCosts / Me.LtTotalSales
1566    End Sub
1567
1568    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1569    ' This method computes the penetration rate per
1570    ' technology for this mfg.
1571    '''
1572    Private Sub ComputeApplicationAndPenetrationRatesPerTechnology(ByVal MY As Integer)
1573        Dim i, j As Integer
1574        Dim nTech As Integer
1575        Dim vIdx As Integer
1576        Dim veh As VehiclesClass
1577        Dim vehRC As Integer
1578        Dim vehSales As Double
1579
1580        nTech = TechnologiesClass.nTotalTechnologies
1581
1582        ' resize overall value
1583        ReDim Me.ApplicationRatePerTechnology(nTech - 1)
1584        ReDim Me.PenetrationRatePerTechnology(nTech - 1)
1585        '
1586        If IsNothing(IW_ApplicationRatePerTechnology) Then
1587            ReDim IW_ApplicationRatePerTechnology(nTech - 1)
1588        End If
1589        '
1590        If IsNothing(IW_PenetrationRatePerTechnology) Then
1591            ReDim IW_PenetrationRatePerTechnology(nTech - 1)
1592        End If
1593        '
1594        ' resize overall light truck value
1595        ReDim Me.LtApplicationRatePerTechnology(nTech - 1)
1596        ReDim Me.LtPenetrationRatePerTechnology(nTech - 1)
1597        '
1598        If IsNothing(IW_LtApplicationRatePerTechnology) Then
```

```
1599            ReDim IW_LtApplicationRatePerTechnology(nTech - 1)
1600        End If
1601        '
1602        If IsNothing(IW_LtPenetrationRatePerTechnology) Then
1603            ReDim IW_LtPenetrationRatePerTechnology(nTech - 1)
1604        End If
1605        '
1606        ' resize by RC values
1607        For i = 0 To nRC - 1
1608            ReDim Me.RcApplicationRatePerTechnology(i)(nTech - 1)
1609            ReDim Me.RcPenetrationRatePerTechnology(i)(nTech - 1)
1610            '
1611            If IsNothing(IW_RcApplicationRatePerTechnology(i)) Then
1612                ReDim IW_RcApplicationRatePerTechnology(i)(nTech - 1)
1613            End If
1614            '
1615            If IsNothing(IW_RcPenetrationRatePerTechnology(i)) Then
1616                ReDim IW_RcPenetrationRatePerTechnology(i)(nTech - 1)
1617            End If
1618        Next i
1619
1620        If Not IsNothing(Me.ValidVehicleIndexes) Then
1621            For i = 0 To nTech - 1
1622                ' reset application and penetration rates for this tech to 0 to avoid over-counting
1623                Me.ApplicationRatePerTechnology(i) = 0
1624                Me.PenetrationRatePerTechnology(i) = 0
1625                '
1626                Me.LtApplicationRatePerTechnology(i) = 0
1627                Me.LtPenetrationRatePerTechnology(i) = 0
1628
1629                ' scan each vehicle for the i-th tech
1630                For j = 0 To Me.nValidVehicles - 1
1631                    vIdx = Me.ValidVehicleIndexes(j)
1632                    veh = CAFE.Vehicles(vIdx)
1633                    If veh.IsValid Then
1634                        vehRC = veh.RegClass
1635                        vehSales = veh.Sales(CAFE.GMYI(MY))
1636
1637                        ' if the j-th vehicle uses the i-th tech for this MY, increment
1638                        ' ApplicationRatePerTechnology counters for that tech
1639                        If veh.TechUsedByMY(CAFE.GMYI(MY))(i) Then
1640                            Me.ApplicationRatePerTechnology(i) += vehSales
1641                            Me.RcApplicationRatePerTechnology(vehRC)(i) += vehSales
1642                            '
1643                            IW_ApplicationRatePerTechnology(i) += vehSales
1644                            IW_RcApplicationRatePerTechnology(vehRC)(i) += vehSales
1645                        End If
1646
1647                        ' if the j-th vehicle uses the i-th tech at all, for any MY,
1648                        ' increment PenetrationRatePerTechnology counters for that tech
1649                        If veh.TechUsed(i) Then
1650                            Me.PenetrationRatePerTechnology(i) += vehSales
1651                            Me.RcPenetrationRatePerTechnology(vehRC)(i) += vehSales
1652                            '
1653                            IW_PenetrationRatePerTechnology(i) += vehSales
1654                            IW_RcPenetrationRatePerTechnology(vehRC)(i) += vehSales
1655                        End If
1656                    End If
1657                Next j
1658
1659                ' calculate light truck rates (using individual rc totals)
1660                For j = 3 To nRC - 1
1661                    Me.LtApplicationRatePerTechnology(i) += Me.RcApplicationRatePerTechnology(j)(i)
1662                    Me.LtPenetrationRatePerTechnology(i) += Me.RcPenetrationRatePerTechnology(j)(i)
1663                    '
1664                    IW_LtApplicationRatePerTechnology(i) += Me.RcApplicationRatePerTechnology(j)(i)
```

```
1665                    IW_LtPenetrationRatePerTechnology(i) += Me.
      RcPenetrationRatePerTechnology(j)(i)
1666                Next j
1667
1668                ' after all vehicles were scaned, divide ApplicationRatePerTechnology
1669                ' counters for this tech by nVeh to get the fraction/percentage of
1670                ' vehicles that use this technology for the given MY
1671                '
1672                ' also, do the same for PenetrationRatePerTechnology counters
1673                '
1674                ' don't calculate for industry wide variables yet, they will be calculated
1675                ' in the ComputeIndustryWideTotalsAndAverages() sub
1676                If Me.TotalSales <> 0 Then
1677                    Me.ApplicationRatePerTechnology(i) /= Me.TotalSales
1678                    Me.PenetrationRatePerTechnology(i) /= Me.TotalSales
1679                End If
1680                '
1681                If Me.LtTotalSales <> 0 Then
1682                    Me.LtApplicationRatePerTechnology(i) /= Me.LtTotalSales
1683                    Me.LtPenetrationRatePerTechnology(i) /= Me.LtTotalSales
1684                End If
1685                '
1686                For j = 0 To nRC - 1
1687                    If Me.RcTotalSales(j) <> 0 Then
1688                        Me.RcApplicationRatePerTechnology(j)(i) /= Me.RcTotalSales(j)
1689                        Me.RcPenetrationRatePerTechnology(j)(i) /= Me.RcTotalSales(j)
1690                    End If
1691                Next j
1692            Next i
1693        End If
1694    End Sub
1695
1696 #End Region
1697
1698
1699 #Region " Methods for manipulating the RC()() array. "
1700
1701    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1702    ' Adds a given value (val) at a given RC index (rc)
1703    '
1704    ' rc:   the index of the RC array, where to add a new value.
1705    ' val:  the new value to add to Me.RC(rc).
1706    '
1707    ' Throws: System.IndexOutOfRangeException if specified rc
1708    '         index is out of range; eg rc < 0 or rc > nRC.
1709    '''
1710    Public Sub RC_Add(ByVal rc As Integer, ByVal val As Integer)
1711        If DBUG Then
1712            denter(Me, "RC_Add(rc, val)")
1713        End If
1714
1715        ' verify for valid rc index
1716        If Not RC_Verify_Index(rc) Then
1717            Return
1718        End If
1719
1720        ' only add the new val if it hasn't been added yet
1721        If RC_Find(rc, val) = -1 Then
1722            ' change the size of Me.RC at index rc
1723            If IsNothing(Me.RC(rc)) Then
1724                ReDim Preserve Me.RC(rc)(0)
1725            Else
1726                ReDim Preserve Me.RC(rc)(Me.RC(rc).Length)
1727            End If
1728
1729            ' add val to RC(rc)
1730            Me.RC(rc)(Me.RC(rc).Length - 1) = val
1731
1732        End If
1733    End Sub
```

```
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Removes a given value (val) from a given RC index (rc)
    '
    ' rc:   the index of the RC array, from where to remove a
    '       given value.
    ' val:  the value to remove from Me.RC(rc).
    '
    ' Returns: true if the specified value was found and removed,
    '          false otherwise.
    '
    ' Throws:  System.IndexOutOfRangeException if specified rc
    '          index is out of range; eg rc < 0 or rc > nRC.
    '''
    Public Function RC_Remove(ByVal rc As Integer, ByVal val As Integer) As Boolean
        If DBUG Then
            denter(Me, "RC_Remove(rc, val)")
        End If

        Dim index As Integer = RC_Find(rc, val)

        If index = -1 Then
            ' val not found in Me.RC(rc), return false
            Return False
        Else
            ' val found at index...
            ' move everything starting with index up by one
            Dim j As Integer
            For j = index To Me.RC(rc).Length - 2
                Me.RC(rc)(j) = Me.RC(rc)(j + 1)
            Next j

            ' clear out the last element by resizing the array
            ReDim Preserve Me.RC(rc)(Me.RC(rc).Length - 2)
            ' exit function
            Return True
        End If

        ' safeguard...
        Return False
    End Function

    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Clears, removes all elements from, an RC array at a given
    ' RC index (rc)
    '
    ' rc:   the index of the RC array to clear
    '
    ' Returns: true if the specified RC array was found and
    '          cleared, false otherwise.
    '
    ' Throws:  System.IndexOutOfRangeException if specified rc
    '          index is out of range; eg rc < 0 or rc > nRC.
    '''
    Public Function RC_Clear(ByVal rc As Integer) As Boolean
        If DBUG Then
            denter(Me, "RC_Clear(rc)")
        End If

        ' verify for valid rc index
        If Not RC_Verify_Index(rc) Then
            Return False
        End If

        'Me.RC(rc) = Nothing
        ReDim Me.RC(rc)(-1)
    End Function

    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Clears, removes all elements from, all RC arrays
```

```
1804    '''
1805    Public Sub RC_Clear()
1806        If DBUG Then
1807            denter(Me, "RC_Clear()")
1808        End If
1809
1810        Dim i As Integer
1811
1812        For i = 0 To Me.nRC - 1
1813            Me.RC(i) = Nothing
1814        Next i
1815    End Sub
1816
1817    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1818    ' Searches for a given value (val) at a given RC index (rc).
1819    '
1820    ' rc:   the index of the RC array, where to search for a
1821    '       given value.
1822    ' val:  the value to search for in Me.RC(rc).
1823    '
1824    ' Returns: -1 if the target RC array is nothing, has no
1825    '          elements (i.e. empty), or if the specified value
1826    '          is not found; otherwise the index of the value is
1827    '          returned.
1828    '
1829    ' Throws:  System.IndexOutOfRangeException if specified rc
1830    '          index is out of range; eg rc < 0 or rc > nRC.
1831    '''
1832    Public Function RC_Find(ByVal rc As Integer, ByVal val As Integer) As Integer
1833        If DBUG Then
1834            denter(Me, "RC_Find(rc, val)")
1835        End If
1836
1837        ' verify for valid rc index
1838        If Not RC_Verify_Index(rc) Then
1839            Return -1
1840        End If
1841
1842        ' verify Me.RC(rc) is not nothing
1843        If IsNothing(Me.RC(rc)) Then
1844            ' array is empty or has not been initialized, return -1
1845            Return -1
1846        End If
1847
1848        Dim i As Integer
1849
1850        ' loop through the Me.RC array at index rc to find 'val'
1851        'For i = 0 To Me.RC(rc).Length - 1
1852        '    If Me.RC(rc)(i) = val Then
1853        '        ' match found, return i (the index of val in the array)
1854        '        Return i
1855        '    End If
1856        'Next i
1857        i = Array.IndexOf(Me.RC(rc), val)
1858        Return i
1859
1860        ' val not found in the given array, return -1
1861        'Return -1
1862    End Function
1863
1864    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1865    ' Moves a given value (val) from a given RC index (fromRC) to
1866    ' a given RC index (toRC)
1867    '
1868    ' fromRC:   the index of the RC array, from where to get the
1869    '           given value.
1870    ' toRC:     the index of the RC array, where to add the given
1871    '           value.
1872    ' val:      the new value to move to Me.RC(rc).
1873    '
```

```
1874    ' Returns: true if the specified value was moved successfuly,
1875    '          false otherwise.
1876    '
1877    ' Throws:  System.IndexOutOfRangeException if specified rc
1878    '          index is out of range; eg rc < 0 or rc > nRC.
1879    ' Throws:  System.ArgumentException if fromRC = toRC.
1880    '          (Optional)
1881    '''
1882    Public Function RC_Move(ByVal fromRC As Integer, ByVal toRC As Integer, ByVal val As
        Integer) As Boolean
1883        If DBUG Then
1884            denter(Me, "RC_Move(fromRC, toRC, val)")
1885        End If
1886
1887        'If fromRC = toRC Then
1888        '    Throw New ArgumentException("The source and destination RC array indeces " &
1889        '                                "were the same (" & toRC & ").  Moving values " &
1890        '                                "within a single RC array is not allowed.")
1891        '    Return False
1892        'End If
1893
1894        If Not RC_Remove(fromRC, val) Then
1895            ' if could not remove, return false
1896            ' (possibly val is not in fromRC)
1897            Return False
1898        Else
1899            ' add, if remove was successful
1900            RC_Add(toRC, val)
1901            RC_Sort(toRC)
1902        End If
1903
1904        Return True
1905    End Function
1906
1907    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1908    ' Gets the length of a given RC index (rc)
1909    '
1910    ' rc:   the index of the RC array for which to get the length.
1911    '
1912    ' Returns: 0, if the target array has no elements (empty) or
1913    '          is nothing, otherwise returns the length of the
1914    '          array.
1915    '
1916    ' Throws:  System.IndexOutOfRangeException if specified rc
1917    '          index is out of range; eg rc < 0 or rc > nRC.
1918    '''
1919    Public Function RC_Length(ByVal rc As Integer) As Integer
1920        If DBUG Then
1921            denter(Me, "RC_Move(rc)")
1922        End If
1923
1924        ' verify for valid rc index
1925        If Not RC_Verify_Index(rc) Then
1926            Return 0
1927        End If
1928
1929        ' verify Me.RC(rc) is not nothing
1930        If IsNothing(Me.RC(rc)) Then
1931            Return 0
1932        End If
1933
1934        Return Me.RC(rc).Length
1935    End Function
1936
1937    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1938    ' Verifies that a given RC index (rc) is not out of range.
1939    '
1940    ' rc:   the index of the RC array to verify.
```

```
1941        '
1942        ' Returns: true if rc is not out of range.
1943        '
1944        ' Throws:  System.IndexOutOfRangeException if specified rc
1945        '          index is out of range; eg rc < 0 or rc > nRC.
1946        '''
1947        Private Function RC_Verify_Index(ByVal rc As Integer) As Boolean
1948            If DBUG Then
1949                denter(Me, "RC_Verify_Index(rc)")
1950            End If
1951
1952            If rc > Me.nRC - 1 Or rc < 0 Then
1953                Throw New System.IndexOutOfRangeException("The RC array index must be " &
1954                                                          "between 0 and " & Me.nRC - 1 & ".")
1955                Return False
1956            End If
1957
1958            Return True
1959        End Function
1960
1961        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
1962        ' Sort an RC list
1963        '
1964        ' Sorts the RC list to make find work faster
1965        ' RC value of -1 means sort all
1966        '''
1967        Public Sub RC_Sort(ByVal rc As Integer)
1968            If rc > 0 And rc < nRC Then
1969                If Not IsNothing(Me.RC(rc)) Then
1970                    Array.Sort(Me.RC(rc))
1971                End If
1972            ElseIf rc = -1 Then
1973                For iRC As Integer = 0 To nRC - 1
1974                    If Not IsNothing(Me.RC(iRC)) Then
1975                        Array.Sort(Me.RC(iRC))
1976                    End If
1977                Next
1978            End If
1979        End Sub
1980
1981    #End Region
1982
1983
1984    #Region " The ""Add"" methods (veh, nameplate, eng, tran) "
1985
1986        Public Sub AddVehicle(ByVal VehIdx As Integer, ByVal MfgIdx As Integer)
1987            If DBUG Then
1988                denter(Me, "AddVehicle(MY, VehIdx)")
1989            End If
1990
1991            ' get the model of the vehicle
1992            Dim Model As String = CAFE.MasterVehicles(VehIdx).Model
1993            ' add this model to the list of nameplates
1994            AddNamePlate(Model, VehIdx, MfgIdx)
1995
1996            ' set the MFgIdx for this vehicle
1997            CAFE.MasterVehicles(VehIdx).MfgIdx = MfgIdx
1998
1999            ' Add to the "All" list
2000            ReDim Preserve Me.AllVehicleIndexes(Me.nAllVehicles)
2001            Me.AllVehicleIndexes(Me.nAllVehicles) = VehIdx
2002            Me.nAllVehicles += 1
2003        End Sub
2004
2005        Private Sub AddNamePlate(ByVal Name As String, ByVal VehIdx As Integer, ByVal MfgIdx
            As Integer)
2006            If DBUG Then
2007                denter(Me, "AddNamePlate(Name, VehIdx)")
2008            End If
2009
```

```
2010            ' first check if a nameplate with this name already exists
2011            Dim index As Integer = FindNamePlate(Name)
2012
2013            If index = -1 Then
2014                ' nameplate name not found
2015                ' creat new instance of the NamePlateClass
2016                Dim NewNamePlate As New NamePlateClass
2017                NewNamePlate.Name = Name
2018                NewNamePlate.MyIndexNumber = nNamePlates
2019                NewNamePlate.MfgIdx = MfgIdx
2020                NewNamePlate.AddUser(VehIdx)
2021                ' add the new nameplate to the Me.NamePlates() array
2022                ReDim Preserve Me.NamePlates(nNamePlates)
2023                Me.NamePlates(nNamePlates) = NewNamePlate
2024                ReDim Preserve Me.NamePlateIndexes(nNamePlates)
2025                Me.NamePlateIndexes(nNamePlates) = nNamePlates
2026                CAFE.MasterVehicles(VehIdx).NamePlateIdx = nNamePlates
2027                nNamePlates += 1
2028            Else
2029                ' nameplate name found, add new vehicle at the found index
2030                Me.NamePlates(index).AddUser(VehIdx)
2031                CAFE.MasterVehicles(VehIdx).NamePlateIdx = index
2032            End If
2033        End Sub
2034
2035        ' returns:  index of the nameplate with the given name,
2036        '           or -1 if name not found.
2037        Private Function FindNamePlate(ByVal Name As String)
2038            If DBUG Then
2039                denter(Me, "FindNamePlate(Name)")
2040            End If
2041
2042            Dim i As Integer
2043
2044            For i = 0 To nNamePlates - 1
2045                If Me.NamePlates(i).Name.Equals(Name) Then
2046                    ' match found, return the index
2047                    Return i
2048                End If
2049            Next i
2050
2051            ' match not found, return -1
2052            Return -1
2053        End Function
2054
2055        Public Sub AddEngine(ByVal EngIdx As Integer, ByVal MfgIdx As Integer)
2056            If DBUG Then
2057                denter(Me, "AddEngine(EngIdx)")
2058            End If
2059
2060            ReDim Preserve EngineIndexes(nEngines)
2061            EngineIndexes(nEngines) = EngIdx
2062            CAFE.MasterEngines(EngIdx).MfgIdx = MfgIdx
2063            nEngines += 1
2064        End Sub
2065
2066        Public Sub AddTransmission(ByVal TransmissionIdx As Integer, ByVal MfgIdx As Integer)
2067            If DBUG Then
2068                denter(Me, "AddTransmission(TransmissionIdx)")
2069            End If
2070
2071            ReDim Preserve TransmissionIndexes(nTransmissions)
2072            TransmissionIndexes(nTransmissions) = TransmissionIdx
2073            CAFE.MasterTransmissions(TransmissionIdx).mfgidx = MfgIdx
2074            nTransmissions += 1
2075        End Sub
2076
2077    #End Region
2078
2079
```

```
2080 #Region " CrossReference and Validate methods "
2081
2082     Public Sub CrossReference()
2083         If DBUG Then
2084             denter(Me, "CrossReference()")
2085         End If
2086
2087         ' - Rectify the references to engines and transmissions.
2088         '   As they exist in the DB, they represent MFG specific references, these indexes
2089         '   must be changed to represent the DB as a whole
2090         ' - Then develop a list of vehicles which use each engine and transmission.
2091
2092         ' cross-reference all vehicles
2093         CrossReference(Me.nAllVehicles, Me.AllVehicleIndexes)
2094
2095     End Sub ' crossreference
2096
2097     Private Sub CrossReference(ByVal nVehicle As Integer, ByRef VehicleIndexes() As
        Integer)
2098         If DBUG Then
2099             denter(Me, "CrossReference(nVehicle, VehicleIndexes())")
2100         End If
2101
2102         Dim v, e, p, t As Integer  ' vehicle, engines, predecessors, and transmitions
        counters
2103
2104         For v = 0 To nVehicle - 1
2105             Dim vIdx As Integer = VehicleIndexes(v)
2106             For e = 0 To nEngines - 1
2107                 If CAFE.MasterVehicles(vIdx).EngineCode = CAFE.MasterEngines(EngineIndexes
        (e)).Engine_Code Then
2108                     ' Got a match
2109                     CAFE.MasterVehicles(vIdx).TrueEngineCode = EngineIndexes(e)
2110                     ' Update the list for this engine to add this vehicle as a user
2111                     CAFE.MasterEngines(EngineIndexes(e)).AddUser(vIdx)
2112                     Exit For
2113                 End If
2114             Next e
2115
2116             For t = 0 To nTransmissions - 1
2117                 If CAFE.MasterVehicles(vIdx).TransmissionCode = CAFE.MasterTransmissions
        (TransmissionIndexes(t)).Transmission_Code Then
2118                     ' Got a match
2119                     CAFE.MasterVehicles(vIdx).TrueTransmissionCode = TransmissionIndexes
        (t)
2120                     ' Update the list for this transmission to add this vehicle as a user
2121                     CAFE.MasterTransmissions(TransmissionIndexes(t)).AddUser(vIdx)
2122                     Exit For
2123                 End If
2124             Next t
2125
2126             If IsNothing(CAFE.MasterVehicles(vIdx).PredecessorFromFile) = False AndAlso
        CAFE.MasterVehicles(vIdx).PredecessorFromFile >= 0 Then
2127                 CAFE.MasterVehicles(vIdx).Predecessor = -1
2128                 For p = 0 To CAFE.MasterVehicles.Length - 1
2129                     If CAFE.MasterVehicles(vIdx).PredecessorFromFile = CAFE.MasterVehicles
        (p).Number Then
2130                         ' Got a match!
2131                         CAFE.MasterVehicles(vIdx).Predecessor = p
2132                         Exit For
2133                     End If
2134                 Next p
2135             End If
2136
2137             CAFE.MasterVehicles(vIdx).ValidateEngineAndTransmission()
2138         Next v
2139     End Sub
2140
2141     Public Sub SplitValidVehicles()
```

```
2142        Dim i As Integer
2143        Dim vIdx As Integer
2144
2145        For i = 0 To nAllVehicles - 1
2146            vIdx = AllVehicleIndexes(i)
2147
2148            If CAFE.MasterVehicles(vIdx).ValidOverAll Then
2149                ' Add to the "valid" list
2150                ReDim Preserve Me.ValidVehicleIndexes(Me.nValidVehicles)
2151                Me.ValidVehicleIndexes(Me.nValidVehicles) = vIdx
2152                Me.nValidVehicles += 1
2153                ManufacturerClass.nValidVehiclesForAllMfgs += 1
2154            Else
2155                ' Add to the "discarded" list
2156                ReDim Preserve Me.DiscardedVehicleIndexes(Me.nDiscardedVehicles)
2157                Me.DiscardedVehicleIndexes(Me.nDiscardedVehicles) = vIdx
2158                Me.nDiscardedVehicles += 1
2159                ManufacturerClass.nDiscardedVehiclesForAllMfgs += 1
2160            End If
2161        Next i
2162    End Sub
2163
2164    Public Function Validate() As Boolean
2165        Me.Valid = Not (IsNothing(Me.ValidVehicleIndexes) OrElse Me.ValidVehicleIndexes.
      Length <= 0)
2166        Return Me.Valid
2167    End Function
2168
2169 #End Region
2170
2171
2172 #Region " CalcCAFE - methods for calculating CAFE level "
2173
2174    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2175    ' Calculates the CAFE levels for this manufacturer by RC
2176    ' for the specified model year.
2177    '''
2178    Public Sub CalcCAFE(ByVal MY As Integer, _
2179                        Optional ByVal ForceReCalc As Boolean = True)
2180        If DBUG Then
2181            denter(Me, "CalcCAFE(MY, [ForceReCalc = True])")
2182        End If
2183
2184        ' counters
2185        Dim i, j As Integer
2186
2187        ' stores the CAFE value for a single RC
2188        Dim RC_CAFE As Double
2189        ' stores total amount of sales for one RC
2190        Dim RC_TotalSales As Double
2191        ' stores the sum of (Sales_i / FE_i) for one RC
2192        Dim RC_SalesOverFE As Double
2193
2194        ' there are 11 rc's--1 unregulated, 2 auto, and 8 lt;
2195        ' scan each rc one at a time and compute the CAFE value
2196        For i = 0 To nRC - 1
2197            ' before proceeding to calculate the current RC,
2198            ' reset all values to clear out previous calculations
2199            RC_CAFE = 0
2200            RC_TotalSales = 0
2201            RC_SalesOverFE = 0
2202
2203            ' it is possible that any one regulatory class may not
2204            ' have any vehicles in it
2205            If Not IsNothing(RC(i)) Then
2206                ' if there are vehicles at the current RC, scan
2207                ' each regulatory class, RC(i)
2208                For j = 0 To RC(i).Length - 1
2209                    ' calculate TotalSales and Sales/FE for this RC
2210                    CalcCAFE_ProcessSingleVehicle(MY, RC(i)(j), RC_TotalSales,
```

```
    RC_SalesOverFE)
2211                Next j
2212
2213                ' calculate the CAFE level for the current RC
2214                'RC_CAFE = Math.Round((RC_TotalSales / RC_SalesOverFE), 1)
2215                RC_CAFE = RC_TotalSales / RC_SalesOverFE
2216            End If
2217
2218            ' save the RC_TotalSales and the total RC_SalesOverFE into global arrays
2219            Me.RcTotalSales(i) = RC_TotalSales
2220            Me.RcSalesOverFE(i) = RC_SalesOverFE
2221            ' check to see if the original total sales has been set
2222            If Double.IsNaN(Me.RcTotalSales_Orig(i)) Then
2223                ' if it hasn't, save it as well
2224                Me.RcTotalSales_Orig(i) = RC_TotalSales
2225            End If
2226            ' check to see if the original sales/fe has been set
2227            If Double.IsNaN(Me.RcSalesOverFE_Orig(i)) Then
2228                ' if it hasn't, save it as well
2229                Me.RcSalesOverFE_Orig(i) = RC_SalesOverFE
2230            End If
2231
2232            ' save the calculated CAFE level for this RC into global array
2233            Me.RcCAFE(i) = RC_CAFE
2234            ' check to see if the original cafe has been set
2235            If Double.IsNaN(Me.RcCAFE_Orig(i)) Then
2236                ' if it hasn't, save it as well
2237                Me.RcCAFE_Orig(i) = RC_CAFE
2238            End If
2239        Next i
2240    End Sub ' CalcCAFE
2241
2242    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2243    ' See ReCalcCAFE().
2244    '''
2245    'Public Function CalcCAFE(ByVal MY As Integer, _
2246    '                         ByVal nTestVeh As Integer, _
2247    '                         ByVal TestVehIdx() As Integer, _
2248    '                         ByVal TestVehFE() As Double, _
2249    '                         Optional ByVal ForceReCalc As Boolean = True) As Double()
2250    '    If DBUG Then
2251    '        denter(Me, "CalcCAFE(MY, nTestVeh, TestVehIdx(), TestVehFE(), [ForceReCalc =
    True])")
2252    '    End If
2253
2254    '    ' call ReCalcCAFE() method
2255    '    Return ReCalcCAFE(MY, nTestVeh, TestVehIdx, TestVehFE, ForceReCalc)
2256    'End Function
2257
2258    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2259    ' Recalculates the CAFE levels for this manufacturer by
2260    ' RC for the specified model year and returns an array of
2261    ' new RC CAFE values. The new values are not stored in
2262    ' this manufacturer and are discarded when the method
2263    ' returns.
2264    '''
2265    Public Function ReCalcCAFE(ByVal MY As Integer, _
2266                               ByVal nTestVeh As Integer, _
2267                               ByVal TestVehIdx() As Integer, _
2268                               ByVal TestVehFE() As Double, _
2269                               Optional ByVal ForceReCalc As Boolean = True, _
2270                               Optional ByVal OverRideSalesOverFE() As Double = Nothing)
    As Double()
2271        If DBUG Then
2272            denter(Me, "ReCalcCAFE(MY, nTestVeh, TestVehIdx(), TestVehFE(), [ForceReCalc =
     True])")
2273        End If
2274
2275        ' counters
2276        Dim i As Integer
```

```
2277
2278        Dim RC_CAFE As Double
2279
2280        ' copy over previous CAFE values for all RCs from RcCAFE
2281        For i = 0 To nRC - 1
2282            Me.RcCAFE_Temp(i) = Me.RcCAFE(i)
2283            If IsNothing(OverRideSalesOverFE) Then
2284                Me.RcSalesOverFE_Temp(i) = Me.RcSalesOverFE(i)
2285            Else
2286                Me.RcSalesOverFE_Temp(i) = OverRideSalesOverFE(i)
2287            End If
2288        Next i
2289
2290        ' scan each test vehicle to get new CAFE values for some (or all) RCs
2291        For i = 0 To nTestVeh - 1
2292            ' get the test vehicle index
2293            Dim vIdx As Integer = TestVehIdx(i)
2294            ' get the regulatory class of the test vehicle
2295            Dim rc As Integer = CAFE.Vehicles(vIdx).RegClass
2296
2297            ' stores the difference between the previous Sales / FE
2298            ' value and the new Sales / FE value, which is calculated
2299            ' using the TestVehFE(i) feul economy
2300            Dim SalesOverFE_Difference As Double
2301            SalesOverFE_Difference = (CAFE.Vehicles(vIdx).Sales(CAFE.GMYI(MY)) / TestVehFE(i)) _
2302                                     - CAFE.Vehicles(vIdx).SalesOverFE
2303            Me.RcSalesOverFE_Temp(rc) += SalesOverFE_Difference
2304        Next i
2305
2306        ' Now loop through RCs and calc CAFE for output
2307        For i = 0 To nRC - 1
2308            Me.RcCAFE_Temp(i) = Me.RcTotalSales(i) / Me.RcSalesOverFE_Temp(i)
2309        Next i
2310
2311        Return Me.RcCAFE_Temp
2312    End Function ' ReCalcCAFE
2313
2314    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2315    ' Alternative entry w/single vehicle
2316    '''
2317    'Public Function ReCalcCAFE(ByVal MY As Integer, _
2318    '                           ByVal TestVehIdx As Integer, _
2319    '                           ByVal TestVehFE As Double, _
2320    '                           Optional ByVal ForceReCalc As Boolean = True) As Double()
2321    '    If DBUG Then
2322    '        denter(Me, "ReCalcCAFE(MY, TestVehIdx, TestVehFE, [ForceReCalc = True])")
2323    '    End If
2324    '
2325    '    ' Make an array for TestVehIdx and TestVehFE and call multiple-veh method
2326    '    Dim VehArray(0) As Integer
2327    '    Dim FEArray(0) As Double
2328    '    VehArray(0) = TestVehIdx
2329    '    FEArray(0) = TestVehFE
2330    '    Return ReCalcCAFE(MY, 1, VehArray, FEArray, ForceReCalc)
2331    'End Function
2332
2333    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2334    ' Calculates the TotalSales and SalesOverFE for a single
2335    ' vehicle specified by vIdx for the specified model year.
2336    '''
2337    Private Sub CalcCAFE_ProcessSingleVehicle(ByVal MY As Integer, _
2338                                              ByVal vIdx As Integer, _
2339                                              ByRef TotalSales As Double, _
2340                                              ByRef SalesOverFE As Double)
2341        If DBUG Then
2342            denter(Me, "CalcCAFE_ProcessSingleVehicle(MY, vIdx, TotalSales, SalesOverFE)")
2343        End If
2344
2345        ' Calculates TotalSales and SalesOverFE for the vehicle at the given index
```

```
2346        Dim Sales_i As Double
2347        Dim FE_i As Double
2348        Dim Sales_iOverFE_i As Double
2349
2350        ' get Sales_i value for the current vehicle for the specified MY
2351        Sales_i = CAFE.Vehicles(vIdx).Sales(CAFE.GMYI(MY))
2352        ' get FE_i value for the current vehicle
2353        FE_i = Math.Round(CAFE.Vehicles(vIdx).FuelEconomy, 1)
2354
2355        ' calculate total sales and Sales_i / FE_i
2356        TotalSales += Sales_i
2357        Sales_iOverFE_i = (Sales_i / FE_i)
2358        SalesOverFE += Sales_iOverFE_i
2359        ' save Sales_i / FE_i for that vehicle
2360        CAFE.Vehicles(vIdx).SalesOverFE = Sales_iOverFE_i
2361    End Sub
2362
2363 #End Region
2364
2365
2366 #Region " CalcCredit - methods for calculating credits "
2367
2368    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2369    ' Calculates the credit for this manufacturer by RC for
2370    ' the specified model year.
2371    '''
2372    Public Sub CalcCredit(ByVal Scenario As Integer, _
2373                          ByVal MY As Integer, _
2374                          Optional ByVal ForceReCalc As Boolean = True, _
2375                          Optional ByVal ReCalcTotalSales As Boolean = True)
2376        If DBUG Then
2377            denter(Me, "CalcCredit(MY, [ForceReCalc = True])")
2378        End If
2379
2380        ' counters
2381        Dim i, j As Integer
2382
2383        ' stores the credit for a single RC
2384        Dim RC_Credit As Double
2385        ' stores total amount of sales for one RC
2386        Dim RC_TotalSales As Double
2387
2388        ' there are 11 rc's--1 unregulated, 2 auto, and 8 lt;
2389        ' scan each rc one at a time and compute the credit
2390        For i = 0 To nRC - 1
2391            ' before proceeding to calculate the current RC,
2392            ' reset all values to clear out previous calculations
2393            RC_Credit = Double.NaN
2394            RC_TotalSales = 0
2395
2396            ' it is possible that any one regulatory class may not
2397            ' have any vehicles in it
2398            If Not IsNothing(RC(i)) Then
2399                ' if there are vehicles at the current RC, scan
2400                ' each regulatory class, RC(i)
2401                If ReCalcTotalSales Then
2402                    For j = 0 To RC(i).Length - 1
2403                        ' calculate TotalSales for this RC
2404                        RC_TotalSales += CAFE.Vehicles(RC(i)(j)).Sales(CAFE.GMYI(MY))
2405                    Next j
2406                Else
2407                    RC_TotalSales = Me.RcTotalSales(i)
2408                End If
2409
2410                ' calculate the credit for the current RC -- Credit = CAFE - STD
2411                RC_Credit = RC_TotalSales * (Me.RcCAFE(i) - Me.CalcSTD(Scenario, MY, i))
2412
2413                ' save the calculated credit for this RC into global array
2414                Me.RcCredit(i) = RC_Credit
2415                ' check to see if the original credit has been set
```

```
2416                If Double.IsNaN(Me.RcCredit_Orig(i)) Then
2417                    ' if it hasn't, save it as well
2418                    Me.RcCredit_Orig(i) = RC_Credit
2419                End If
2420            End If
2421        Next i
2422    End Sub ' CalcCredit
2423
2424    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2425    ' See ReCalcCredit().
2426    '''
2427    Public Function CalcCredit(ByVal Scenario As Integer, _
2428                               ByVal MY As Integer, _
2429                               ByVal nTestVeh As Integer, _
2430                               ByVal TestVehIdx() As Integer, _
2431                               ByVal TestVehFE() As Double, _
2432                               ByVal TestVehArea() As Double, _
2433                               ByVal TestVehWeight() As Double, _
2434                               Optional ByRef TestCAFE() As Double = Nothing, _
2435                               Optional ByVal ForceReCalc As Boolean = True) As Double()
2436        If DBUG Then
2437            denter(Me, "CalcCredit(MY, nTestVeh, TestVehIdx(), TestVehFE(), TestVehArea(), ↙
     TestVehWeight(), [ForceReCalc = True])")
2438        End If
2439
2440        ' call ReCalcCredit() method
2441        Return ReCalcCredit(Scenario, MY, TestVehIdx, TestVehFE, TestVehArea, _
2442                TestVehWeight, TestCAFE, ForceReCalc)
2443    End Function
2444
2445    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2446    ' Recalculates the credits for this manufacturer by RC
2447    ' for the specified model year and returns an array of
2448    ' new RC credit values. The new values are not stored
2449    ' in this manufacturer and are discarded when the method
2450    ' returns.
2451    '''
2452    Public Function ReCalcCredit(ByVal Scenario As Integer, _
2453                                 ByVal MY As Integer, _
2454                                 ByVal TestVehIdx() As Integer, _
2455                                 ByVal TestVehFE() As Double, _
2456                                 ByVal TestVehArea() As Double, _
2457                                 ByVal TestVehWeight() As Double, _
2458                                 Optional ByRef TestCAFE() As Double = Nothing, _
2459                                 Optional ByVal ForceReCalc As Boolean = True) As Double()
2460        If DBUG Then
2461            denter(Me, "ReCalcCredit(MY, nTestVeh, TestVehIdx(), TestVehFE(), TestVehArea ↙
    (), TestVehWeight(), [ForceReCalc = True])")
2462        End If
2463
2464        ' counters
2465        Dim nTestVeh = TestVehIdx.Length
2466        Dim i As Integer
2467        Dim RC_Credit As Double
2468
2469        Dim TestRC() As Integer
2470
2471        ' Scan list of vehicles: If any have had a weight change AND the scenario is      ↙
    weight-based, then
2472        ' we've got work to do
2473        ReDim TestRC(nTestVeh - 1)
2474        Dim OverRideSalesOverFE() As Double
2475
2476        For v As Integer = 0 To nTestVeh - 1
2477            TestRC(v) = CAFE.Standards.GetRegClass(Scenario, MY, TestVehIdx(v),          ↙
    TestVehWeight(v))
2478            If CAFE.Scenarios.Scenarios(Scenario).LtRegClassAttribute.ToUpper = "WEIGHT" ↙
    Then
2479                If CAFE.Vehicles(TestVehIdx(v)).CurbWeight <> TestVehWeight(v) Then
2480                    If TestRC(v) <> CAFE.Vehicles(TestVehIdx(v)).RegClass Then
```

```
2481                        ' Now we must pluck this vehicle out of its' RC and into the possible new RC.
2482                        ' Then we'll recalc CAFE and STD presently.
2483                        ' JVS...
2484                        ' First, if the override array hasn't been declared, then do so
2485                        If IsNothing(OverRideSalesOverFE) Then
2486                            ReDim OverRideSalesOverFE(nRC - 1)
2487                        End If
2488
2489                        ' next, populate it
2490                        Array.Copy(Me.RcSalesOverFE, OverRideSalesOverFE, Me.RcSalesOverFE.Length)
2491
2492                        ' next, modify the former and possible new RCs
2493                        Dim OldRC = CAFE.Vehicles(TestVehIdx(v)).RegClass
2494                        OverRideSalesOverFE(OldRC) -= (CAFE.Vehicles(TestVehIdx(v)).SalesOverFE)
2495                        OverRideSalesOverFE(TestRC(v)) += (CAFE.Vehicles(TestVehIdx(v)).SalesOverFE)
2496                    End If
2497                End If
2498            End If
2499        Next v
2500
2501        ' get new cafe values
2502        Dim NewCafe() As Double
2503        NewCafe = ReCalcCAFE(MY, nTestVeh, TestVehIdx, TestVehFE, True, OverRideSalesOverFE)
2504
2505        If Not IsNothing(TestCAFE) Then
2506            ReDim TestCAFE(NewCafe.Length - 1)
2507            Array.Copy(NewCafe, TestCAFE, NewCafe.Length)
2508        End If
2509
2510        For i = 0 To nRC - 1
2511            ' only consider scanning test vehicles for existent RC's
2512            If Not IsNothing(Me.RC(i)) Then
2513                ' Vehicle list can have multiple RCs, so strip out vehicles an RC at a time
2514                Dim myVeh() As Integer
2515                Dim myFE() As Double
2516                Dim myArea() As Double
2517                Dim myWeight() As Double
2518                Dim myIdx As Integer = 0
2519                For v As Integer = 0 To nTestVeh - 1
2520                    If CAFE.Vehicles(TestVehIdx(v)).RegClass = i Then
2521                        ' Add this vehicle to the list
2522                        ReDim Preserve myVeh(myIdx)
2523                        ReDim Preserve myFE(myIdx)
2524                        ReDim Preserve myArea(myIdx)
2525                        ReDim Preserve myWeight(myIdx)
2526                        myVeh(myIdx) = TestVehIdx(v)
2527                        myFE(myIdx) = TestVehFE(v)
2528                        myArea(myIdx) = TestVehArea(v)
2529                        myWeight(myIdx) = TestVehWeight(v)
2530                        myIdx += 1
2531                    End If
2532                Next v
2533                ' get new std for this rc
2534                Dim mSTD As Double = CalcSTD(Scenario, MY, i, myIdx, myVeh, myArea, myWeight)
2535                ' get new cafe for this rc
2536                Dim mCAFE As Double = NewCafe(i)
2537                ' calculate new credit
2538                RC_Credit = Me.RcTotalSales(i) * (mCAFE - mSTD)
2539                ' save new credit
2540                Me.RcCredit_Temp(i) = RC_Credit
2541            End If
2542        Next i
2543
```

```
            Return Me.RcCredit_Temp
        End Function ' ReCalcCredit

#End Region


#Region " CalcFines - methods for calculating fines "

        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        ' Calculates the fines for this manufacturer by RC for
        ' specified model year.
        '''
        Public Sub CalcFines(ByVal Scenario As Integer, _
                             ByVal MY As Integer, _
                             Optional ByVal ForceReCalc As Boolean = True)
            If DBUG Then
                denter(Me, "CalcFines(MY, [ForceReCalc = True])")
            End If

            Dim i As Integer

            ' call the overloaded private function to calculate
            ' the fines with the default RC credit
            Me.RcFines = CalcFines(Scenario, MY, Me.RcCredit, ForceReCalc, True)

            For i = 0 To Me.RcFines.Length - 1
                ' check to see if the original fines have been set
                If Double.IsNaN(Me.RcFines_Orig(i)) Then
                    ' if it hasn't, save it as well
                    Me.RcFines_Orig(i) = Me.RcFines(i)
                End If
            Next i
        End Sub

        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        ' Calculates the fines for this manufacturer by RC for
        ' the specified model year with the specified RC credits,
        ' RcCredit(), array. This function returns the calculated
        ' fines as an array by RC.
        '''
        Private Function CalcFines(ByVal Scenario As Integer, _
                                   ByVal MY As Integer, _
                                   ByRef lRcCredit() As Double, _
                                   Optional ByVal ForceReCalc As Boolean = True, _
                                   Optional ByVal SaveFines As Boolean = False) As Double()

            If DBUG Then
                denter(Me, "CalcFines(MY, RcCredit(), [ForceReCalc = True], [SaveFines = False])")
            End If

            ' counters
            Dim i As Integer

            ' stores the total fines for all RC's
            Dim TotalFines As Double
            ' stores the fines for a single RC
            Dim RC_Fines As Double
            ' stores the fines for all RC's
            Dim m_RcFines(nRC - 1) As Double

            ' stores the trading pools
            'Dim TradingPool(4)() As Integer

            ' stores whether LT trading is allowed for this scenario for the given MY
            Dim TradeAllowed As Boolean = CAFE.Scenarios.Scenarios(Scenario).TradeAllowed(CAFE.GMYI(MY))

            ' if trading is allowed for this scenario for the given MY
            ' then set up the LT trading pool to contain all light truck
```

```
2612        ' regulatory classes.
2613        If TradeAllowed Then
2614            ' redim the TradingPool()() array to account for five
2615            ' trading pools: 0 - unregulated,   1 - domestic auto,
2616            '                2 - imported auto, 3 - light trucks
2617            '                5 - heavy light trucks covered as a separate class
2618            ReDim TradingPool(4)
2619            ReDim TradingPool(3)(nRC - 5)
2620            'ReDim TradingPool(4)(nRC - 5)
2621
2622            ' store light truck RC indices into the trading pool
2623            For i = 3 To nRC - 2
2624                TradingPool(3)(i - 3) = i
2625            Next i
2626
2627            ' store other RC indices into the trading pools
2628            For i = 0 To 2
2629                ReDim TradingPool(i)(0)
2630                TradingPool(i)(0) = i
2631            Next i
2632
2633            ' store heavy light trucks as a distinct class into the trading pool
2634            ReDim TradingPool(4)(0)
2635            TradingPool(4)(0) = 11
2636
2637        Else
2638            ' trading is not allowed, redim TradingPool()() array
2639            ' to be length nRC so that it contains one RC per
2640            ' trading pool.
2641            ReDim TradingPool(nRC - 1)
2642
2643            For i = 0 To nRC - 1
2644                ReDim TradingPool(i)(0)
2645                TradingPool(i)(0) = i
2646            Next i
2647        End If
2648
2649        ' scan each trading pool one at a time and compute the fines
2650        For i = 0 To TradingPool.Length - 1
2651            ' before proceeding to calculate the current trading pool,
2652            ' reset all values to clear out previous calculations
2653            RC_Fines = Double.NaN
2654
2655            ' calculate the fines for the current trading pool
2656            RC_Fines = -(CAFE.Parameters.K_f) * SumCreditsInTradingPool(TradingPool(i),
    lRcCredit)
2657
2658            ' save the calculated fines for this RC into array
2659            m_RcFines(i) = RC_Fines
2660        Next i
2661
2662        TotalFines = 0
2663        For i = 0 To TradingPool.Length - 1
2664            If Double.IsInfinity(m_RcFines(i)) = False And Double.IsNaN(m_RcFines(i)) =
    False Then
2665                TotalFines += m_RcFines(i)
2666            End If
2667        Next i
2668
2669        If SaveFines Then
2670            Me.TotalFines = TotalFines
2671            ' check to see if the original total fines have been set
2672            If Double.IsNaN(Me.TotalFines_Orig) Then
2673                ' if it hasn't, save it as well
2674                Me.TotalFines_Orig = TotalFines
2675            End If
2676        End If
2677
2678        ' always save the temp fines
2679        Me.TotalFines_Temp = TotalFines
```

```
2680
2681            ' return the fines array
2682            Return m_RcFines
2683        End Function
2684
2685        '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2686        ' Sums up the credits of all RCs within the specified
2687        ' trading pool and returns the minimum of the sum and 0.
2688        '
2689        ' TradingPool() stores regulatory class (RC) indexes.
2690        '''
2691        Public Function SumCreditsInTradingPool(ByVal TradingPool() As Integer, _
2692                                                ByRef lRcCredit() As Double, _
2693                                                Optional ByVal MaxAtZero As Boolean = True)
        As Double
2694            If DBUG Then
2695                denter(Me, "SumCreditsInTradingPool(TradingPool())")
2696            End If
2697
2698            If IsNothing(TradingPool) Then
2699                ' return 0, if the trading pool does not have
2700                ' any RC indexes in it.
2701                Return 0
2702            End If
2703
2704            Dim i As Integer
2705            Dim rc As Integer       ' stores one rc value from the trading pool
2706            Dim sum As Double = 0   ' stores the sum of credits within this trading pool
2707
2708            ' add up the credits in this trading pool
2709            For i = 0 To TradingPool.Length - 1
2710                rc = TradingPool(i)
2711                sum += lRcCredit(rc)
2712            Next i
2713
2714            If MaxAtZero Then
2715                ' return the minimum of the sum of credits and 0
2716                sum = Math.Min(sum, 0)
2717            End If
2718            Return sum
2719        End Function
2720
2721        '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2722        ' Sums up the minimums of credits and 0 for all RCs not
2723        ' in the specified trading pool and returns the sum.
2724        '
2725        ' TradingPool() stores regulatory class (RC) indexes.
2726        '''
2727        Private Function SumCreditsNotInTradingPool(ByVal TradingPool() As Integer, _
2728                                                    ByRef lRcCredit() As Double) As Double
2729            If DBUG Then
2730                denter(Me, "SumCreditsNotInTradingPool(TradingPool())")
2731            End If
2732
2733            Dim i As Integer
2734            Dim sum As Double = 0   ' stores the sum of credits not in this trading pool
2735
2736            For i = 0 To nRC - 1
2737                If IsNothing(TradingPool) OrElse Array.IndexOf(TradingPool, i) = -1 Then
2738                    ' i-th RC is not in the TradingPool--add it to the sum
2739                    sum += Math.Min(lRcCredit(i), 0)
2740                End If
2741            Next i
2742
2743            ' return the sum of the minimums of credits(i) and 0
2744            Return sum
2745        End Function
2746
2747        '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2748        ' See ReCalcFines().
```

```
2749    '''
2750    Public Function CalcFines(ByVal Scenario As Integer, _
2751                              ByVal MY As Integer, _
2752                              ByVal nTestVeh As Integer, _
2753                              ByVal TestVehIdx() As Integer, _
2754                              ByVal TestVehFE() As Double, _
2755                              ByVal TestVehArea() As Double, _
2756                              ByVal TestVehWeight() As Double, _
2757                              Optional ByVal ForceReCalc As Boolean = True) As Double()
2758        If DBUG Then
2759            denter(Me, "CalcFines(MY, nTestVeh, TestVehIdx(), TestVehFE(), TestVehArea(), ↙
    TestVehWeight(), [ForceReCalc = True])")
2760        End If
2761
2762        ' call ReCalcFines() method
2763        Return ReCalcFines(Scenario, MY, nTestVeh, TestVehIdx, TestVehFE, TestVehArea, ↙
    TestVehWeight, ForceReCalc)
2764    End Function
2765
2766    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2767    ' Recalculates the over-all and by RC fines for this
2768    ' manufacturer for the specified model year and returns
2769    ' an array of new RC fines. The new values are not
2770    ' stored in this manufacturer and are discarded when the
2771    ' method returns.
2772    '''
2773    Public Function ReCalcFines(ByVal Scenario As Integer, _
2774                                ByVal MY As Integer, _
2775                                ByVal nTestVeh As Integer, _
2776                                ByVal TestVehIdx() As Integer, _
2777                                ByVal TestVehFE() As Double, _
2778                                ByVal TestVehArea() As Double, _
2779                                ByVal TestVehWeight() As Double, _
2780                                Optional ByVal ForceReCalc As Boolean = True) As Double()
2781        If DBUG Then
2782            denter(Me, "ReCalcFines(MY, nTestVeh, TestVehIdx(), TestVehFE(), TestVehArea() ↙
    , TestVehWeight(), [ForceReCalc = True])")
2783        End If
2784
2785        ' get new credit values
2786        Dim NewCredit() As Double = ReCalcCredit(Scenario, MY, TestVehIdx, _
2787                                                 TestVehFE, TestVehArea, TestVehWeight)
2788        ' call the overloaded private function to calculate
2789        ' the fines with the newly calculated RC credit
2790        Me.RcFines_Temp = CalcFines(Scenario, MY, NewCredit, ForceReCalc)
2791
2792        Return Me.RcFines_Temp
2793    End Function
2794
2795    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2796    ' Alternative entry w/single vehicle
2797    '''
2798    Public Function ReCalcFines(ByVal Scenario As Integer, _
2799                                ByVal MY As Integer, _
2800                                ByVal TestVehIdx As Integer, _
2801                                ByVal TestVehFE As Double, _
2802                                ByVal TestVehArea As Double, _
2803                                ByVal TestVehWeight As Double, _
2804                                Optional ByVal ForceReCalc As Boolean = True) As Double()
2805        If DBUG Then
2806            denter(Me, "ReCalcFines(MY, TestVehIdx, TestVehFE, TestVehArea, TestVehWeight, ↙
     [ForceReCalc = True])")
2807        End If
2808
2809        ' Make everything into an array and call multi-vehicle method
2810        Dim veh() As Integer = {TestVehIdx}
2811        Dim fe() As Double = {TestVehFE}
2812        Dim area() As Double = {TestVehArea}
2813        Dim weight() As Double = {TestVehWeight}
2814        Return ReCalcFines(Scenario, MY, 1, veh, fe, area, weight, ForceReCalc)
```

```
2815     End Function
2816
2817     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2818     ' Calculate impacts (CAFE and Fines) of applying FE improvements
2819     ' (and possibly weight reductions) to a set of vehicles
2820     '''
2821     Public Sub CalcImpacts(ByVal Scenario As Integer, _
2822                            ByVal MY As Integer, _
2823                            ByVal TestVehIdx() As Integer, _
2824                            ByVal TestVehFE() As Double, _
2825                            ByVal TestVehArea() As Double, _
2826                            ByVal TestVehWeight() As Double, _
2827                            ByRef TestCAFE() As Double, _
2828                            ByRef TestFines() As Double)
2829         If DBUG Then
2830             denter(Me, "ReCalcImpacts(MY, TestVehIdx(), TestVehFE(), TestVehArea(),
     TestVehWeight(), TestCAFE(), TestFines())")
2831         End If
2832
2833         ' get new credit values
2834         Dim NewCredit() As Double
2835         NewCredit = ReCalcCredit(Scenario, MY, TestVehIdx, _
2836                                                 TestVehFE, TestVehArea, TestVehWeight,
     TestCAFE)
2837         ' call the overloaded private function to calculate
2838         ' the fines with the newly calculated RC credit
2839         Me.RcFines_Temp = CalcFines(Scenario, MY, NewCredit)
2840         ReDim TestFines(Me.RcFines_Temp.Length - 1)
2841         Array.Copy(Me.RcFines_Temp, TestFines, Me.RcFines_Temp.Length)
2842
2843     End Sub
2844
2845 #End Region
2846
2847
2848 #Region " CalcSTD - methods for calculating standards "
2849
2850     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2851     ' Calculates the standards for this manufacturer by RC
2852     ' for the specified model year.
2853     '''
2854     Public Function CalcSTD(ByVal Scenario As Integer, _
2855                             ByVal MY As Integer, _
2856                             ByVal RegClass As Integer) As Double
2857         If DBUG Then
2858             denter(Me, "CalcSTD(MY, RegClass)")
2859         End If
2860
2861         Return CAFE.Standards.GetStandard(Scenario, MY, MfgIndex, RegClass)
2862     End Function
2863
2864     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
2865     ' Calculates the standards for this manufacturer by RC
2866     ' for the specified model year.
2867     '''
2868     Public Function CalcSTD(ByVal Scenario As Integer, _
2869                             ByVal MY As Integer, _
2870                             ByVal RegClass As Integer, _
2871                             ByVal nTestVeh As Integer, _
2872                             ByVal TestVehIdx() As Integer, _
2873                             ByVal TestVehArea() As Double, _
2874                             ByVal TestVehWeight() As Double) As Double
2875         If DBUG Then
2876             denter(Me, "CalcSTD(MY, RegClass, nTestVeh, TestVehIdx(), TestVehArea(),
     TestVehWeight())")
2877         End If
2878
2879         Return CAFE.Standards.GetStandard(Scenario, MY, MfgIndex, RegClass, _
2880                                           nTestVeh, TestVehIdx, TestVehArea,
     TestVehWeight)
```

```
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This class stores data that will be outputed in reports
'''
Public Class ManufacturerStackClass
    Inherits CafeStackClass

    ' manufacturer's data at the specific model year
    Public Manufacturer As String
    '
    ' values overall
    Public TotalSales_Overall As Double
    Public AverageFuelEconomy_Overall As Double
    Public AverageCurbWeight_Overall As Double
    Public AverageArea_Overall As Double
    Public AverageTechnologyCostsRPE_Overall As Double
    Public AverageFinesIncurredRPE_Overall As Double
    Public AveragePriceIncrease_Overall As Double
    Public TotalTechnologyCosts_Overall As Double
    Public TotalFinesOwed_Overall As Double
    Public TotalPriceIncrease_Overall As Double
    Public ApplicationRatePerTechnology_Overall() As Double
    Public PenetrationRatePerTechnology_Overall() As Double
    Protected TotalUSEmploymentHours_Overall As Double            ' employment hours is protected because
    Public TotalUSEmploymentJobs_Overall As Double                ' we don't want it to be part of the stack
    ' values light truck overall
    Public TotalSales_LtOverall As Double
    Public AverageFuelEconomy_LtOverall As Double
    Public AverageCurbWeight_LtOverall As Double
    Public AverageArea_LtOverall As Double
    Public AverageTechnologyCostsRPE_LtOverall As Double
    Public AverageFinesIncurredRPE_LtOverall As Double
    Public AveragePriceIncrease_LtOverall As Double
    Public TotalTechnologyCosts_LtOverall As Double
    Public TotalFinesOwed_LtOverall As Double
    Public TotalPriceIncrease_LtOverall As Double
    Public ApplicationRatePerTechnology_LtOverall() As Double
    Public PenetrationRatePerTechnology_LtOverall() As Double
    Protected TotalUSEmploymentHours_LtOverall As Double          ' employment hours is protected because
    Public TotalUSEmploymentJobs_LtOverall As Double              ' we don't want it to be part of the stack
    ' values by RC
    Protected Const nRC As Integer = ManufacturerClass.nRC
    Public TotalSales_ByRC(nRC - 1) As Double
    Public AverageFuelEconomy_ByRC(nRC - 1) As Double
    Public AverageCurbWeight_ByRC(nRC - 1) As Double
    Public AverageArea_ByRC(nRC - 1) As Double
    Public AverageTechnologyCostsRPE_ByRC(nRC - 1) As Double
    Public AverageFinesIncurredRPE_ByRC(nRC - 1) As Double
    Public AveragePriceIncrease_ByRC(nRC - 1) As Double
    Public TotalTechnologyCosts_ByRC(nRC - 1) As Double
    Public TotalFinesOwed_ByRC(nRC - 1) As Double
    Public TotalPriceIncrease_ByRC(nRC - 1) As Double
    Public ApplicationRatePerTechnology_ByRC(nRC - 1)() As Double
    Public PenetrationRatePerTechnology_ByRC(nRC - 1)() As Double
    Protected TotalUSEmploymentHours_ByRC(nRC - 1) As Double     ' employment hours is protected because
    Public TotalUSEmploymentJobs_ByRC(nRC - 1) As Double         ' we don't want it to be part of the stack
    ' values overall and by RC
    Protected Const nTotal As Integer = ManufacturerClass.nRC + 2
    Protected TotalSales(nTotal - 1) As Double                         ' 1  - number, decimal 2 places
    Protected AverageFuelEconomy(nTotal - 1) As Double                 ' 2  -
    Protected AverageCurbWeight(nTotal - 1) As Double                  ' 3  -
    Protected AverageArea(nTotal - 1) As Double                        ' 4  -
    Protected AverageTechnologyCostsRPE(nTotal - 1) As Double          ' 5  - accounting, decimal
```

```
63      Protected AverageFinesIncurredRPE(nTotal - 1) As Double              ' 6  -
64      Protected AveragePriceIncrease(nTotal - 1) As Double                 ' 7  -
65      Protected TotalTechnologyCosts(nTotal - 1) As Double                 ' 8  -
66      Protected TotalFinesOwed(nTotal - 1) As Double                       ' 9  -
67      Protected TotalPriceIncrease(nTotal - 1) As Double                   ' 10 -
68      Protected ApplicationRatePerTechnology(nTotal - 1)() As Double       ' 11 - number,
        decimal 1 place
69      Protected PenetrationRatePerTechnology(nTotal - 1)() As Double       ' 12
70      Protected TotalUSEmploymentHours(nTotal - 1) As Double       ' employment hours is not
        part of the stack
71      Protected TotalUSEmploymentJobs(nTotal - 1) As Double                ' 13 - number
72
73      Public Shared nUsedLTRCs() As Integer           ' by scenario
74      Public Shared nUsedRCs() As Integer             ' by scenario
75      Public Shared UsedRCs()() As Boolean            ' by scenario
76
77      Public Sub New()
78          MyBase.New()
79      End Sub
80
81      '''
82      ' Switch = 0, 1, or 2
83      '        0 = save regular values
84      '        1 = save absolute delta values
85      '        2 = save percent delta values
86      '
87      Public Sub New(ByVal Scenario As Integer, _
88                     ByVal MY As Integer, _
89                     ByVal MfgIdx As Integer, _
90                     Optional ByVal Switch As Integer = 0)
91          '
92          Me.New()
93          CreateStack(Scenario, MY, MfgIdx, Switch)
94      End Sub
95
96      '''
97      ' Sets all values to val (0 or Double.Epsilon), where epsilon indicates that these
        values have
98      ' not been calculated yet.
99      '
100     Private Overloads Sub init(ByVal val As Double)
101         Dim i As Integer
102         Dim nTech As Integer = TechnologiesClass.nTotalTechnologies
103
104         If val <> 0 And val <> Double.Epsilon Then
105             val = 0
106         End If
107
108         Me.TotalSales_Overall = val
109         Me.AverageFuelEconomy_Overall = val
110         Me.AverageCurbWeight_Overall = val
111         Me.AverageArea_Overall = val
112         Me.AverageTechnologyCostsRPE_Overall = val
113         Me.AverageFinesIncurredRPE_Overall = val
114         Me.AveragePriceIncrease_Overall = val
115         Me.TotalTechnologyCosts_Overall = val
116         Me.TotalFinesOwed_Overall = val
117         Me.TotalPriceIncrease_Overall = val
118         ReDim Me.ApplicationRatePerTechnology_Overall(nTech - 1)
119         ReDim Me.PenetrationRatePerTechnology_Overall(nTech - 1)
120         Me.init(Me.ApplicationRatePerTechnology_Overall, val)
121         Me.init(Me.PenetrationRatePerTechnology_Overall, val)
122         Me.TotalUSEmploymentHours_Overall = val
123         Me.TotalUSEmploymentJobs_Overall = val
124
125         Me.TotalSales_LtOverall = val
126         Me.AverageFuelEconomy_LtOverall = val
127         Me.AverageCurbWeight_LtOverall = val
128         Me.AverageArea_LtOverall = val
129         Me.AverageTechnologyCostsRPE_LtOverall = val
```

```
130         Me.AverageFinesIncurredRPE_LtOverall = val
131         Me.AveragePriceIncrease_LtOverall = val
132         Me.TotalTechnologyCosts_LtOverall = val
133         Me.TotalFinesOwed_LtOverall = val
134         Me.TotalPriceIncrease_LtOverall = val
135         ReDim Me.ApplicationRatePerTechnology_LtOverall(nTech - 1)
136         ReDim Me.PenetrationRatePerTechnology_LtOverall(nTech - 1)
137         Me.init(Me.ApplicationRatePerTechnology_LtOverall, val)
138         Me.init(Me.PenetrationRatePerTechnology_LtOverall, val)
139         Me.TotalUSEmploymentHours_LtOverall = val
140         Me.TotalUSEmploymentJobs_LtOverall = val
141
142         Me.init(Me.TotalSales_ByRC, val)
143         Me.init(Me.AverageFuelEconomy_ByRC, val)
144         Me.init(Me.AverageCurbWeight_ByRC, val)
145         Me.init(Me.AverageArea_ByRC, val)
146         Me.init(Me.AverageTechnologyCostsRPE_ByRC, val)
147         Me.init(Me.AverageFinesIncurredRPE_ByRC, val)
148         Me.init(Me.AveragePriceIncrease_ByRC, val)
149         Me.init(Me.TotalTechnologyCosts_ByRC, val)
150         Me.init(Me.TotalFinesOwed_ByRC, val)
151         Me.init(Me.TotalPriceIncrease_ByRC, val)
152         For i = 0 To nRC - 1
153             ReDim Me.ApplicationRatePerTechnology_ByRC(i)(nTech - 1)
154             ReDim Me.PenetrationRatePerTechnology_ByRC(i)(nTech - 1)
155             Me.init(Me.ApplicationRatePerTechnology_ByRC(i), val)
156             Me.init(Me.PenetrationRatePerTechnology_ByRC(i), val)
157         Next i
158         Me.init(Me.TotalUSEmploymentHours_ByRC, val)
159         Me.init(Me.TotalUSEmploymentJobs_ByRC, val)
160     End Sub
161
162     '''
163     ' Switch = 0, 1, or 2
164     '        0 = save regular values
165     '        1 = save absolute delta values
166     '        2 = save percent delta values
167     '
168     Public Function CreateStack(ByVal Scenario As Integer, _
169                                 ByVal MY As Integer, _
170                                 ByVal MfgIdx As Integer, _
171                                 Optional ByVal Switch As Integer = 0) As
        ManufacturerStackClass
172         '
173         Me.MY = MY
174         Me.Scenario = Scenario
175         Me.Manufacturer = CAFE.Mfgs(MfgIdx).MfgName
176
177         Select Case Switch
178             Case 0      ' regular vals
179                 Me.CreateStackReg(MY, MfgIdx)
180             Case 1      ' absolute delta vals
181                 Me.CreateStackDelta(Scenario, MY, MfgIdx, MathOP.Subtract)
182             Case 2      ' % delta vals
183                 Me.CreateStackDelta(Scenario, MY, MfgIdx, MathOP.Divide)
184         End Select
185
186         Me.CreateStackCommon()
187         Me.RemoveNaNs()
188
189         Return Me
190     End Function
191
192     Private Sub CreateStackReg(ByVal MY As Integer, ByVal MfgIdx As Integer)
193         Dim i, j As Integer
194         Dim mfg As ManufacturerClass
195         Dim MYIdx As Integer
196         Dim nTech As Integer
197
198         mfg = CAFE.Mfgs(MfgIdx)                                    ' get mfg
```

```
199         MYIdx = CAFE.GMYI(MY)                                   ' get model year index
200         nTech = TechnologiesClass.nTotalTechnologies            ' get number of techs
201
202         ' values overall
203         Me.TotalSales_Overall = mfg.TotalSales
204         Me.AverageFuelEconomy_Overall = mfg.AverageFuelEconomy
205         Me.AverageCurbWeight_Overall = mfg.AverageCWAfter
206         Me.AverageArea_Overall = mfg.AverageAreaAfter
207         Me.AverageTechnologyCostsRPE_Overall = mfg.AverageTechCosts
208         Me.AverageFinesIncurredRPE_Overall = mfg.AverageFinesOwed
209         Me.AveragePriceIncrease_Overall = mfg.AveragePrice
210         Me.TotalTechnologyCosts_Overall = mfg.TotalTechCosts / 1000000          ' (in
        millions)
211         Me.TotalFinesOwed_Overall = mfg.TotalFines / 1000000                    ' (in
        millions)
212         Me.TotalPriceIncrease_Overall = mfg.TotalPurchaseCosts / 1000000        ' (in
        millions)
213         ReDim Me.ApplicationRatePerTechnology_Overall(nTech - 1)
214         ReDim Me.PenetrationRatePerTechnology_Overall(nTech - 1)
215         For i = 0 To nTech - 1
216             Me.ApplicationRatePerTechnology_Overall(i) = mfg.ApplicationRatePerTechnology
        (i)
217             Me.PenetrationRatePerTechnology_Overall(i) = mfg.PenetrationRatePerTechnology
        (i)
218         Next i
219         Me.TotalUSEmploymentHours_Overall = mfg.TotalUSEmploymentHours / 1000   ' (in
        thousands)
220         Me.TotalUSEmploymentJobs_Overall = mfg.TotalUSEmploymentHours / Me.HoursPerYear
221         ' values light truck overall
222         Me.TotalSales_LtOverall = mfg.LtTotalSales
223         Me.AverageFuelEconomy_LtOverall = mfg.LtAverageFuelEconomy
224         Me.AverageCurbWeight_LtOverall = mfg.LtAverageCurbWeight
225         Me.AverageArea_LtOverall = mfg.LtAverageArea
226         Me.AverageTechnologyCostsRPE_LtOverall = mfg.LtAverageTechCosts
227         Me.AverageFinesIncurredRPE_LtOverall = mfg.LtAverageFinesOwed
228         Me.AveragePriceIncrease_LtOverall = mfg.LtAveragePrice
229         Me.TotalTechnologyCosts_LtOverall = mfg.LtTotalTechCosts / 1000000      '
        (in millions)
230         Me.TotalFinesOwed_LtOverall = mfg.LtTotalFinesOwed / 1000000
         ' (in millions)
231         Me.TotalPriceIncrease_LtOverall = mfg.LtTotalPurchaseCosts / 1000000    '
        (in millions)
232         ReDim Me.ApplicationRatePerTechnology_LtOverall(nTech - 1)
233         ReDim Me.PenetrationRatePerTechnology_LtOverall(nTech - 1)
234         For i = 0 To nTech - 1
235             Me.ApplicationRatePerTechnology_LtOverall(i) = mfg.
        LtApplicationRatePerTechnology(i)
236             Me.PenetrationRatePerTechnology_LtOverall(i) = mfg.
        LtPenetrationRatePerTechnology(i)
237         Next i
238         Me.TotalUSEmploymentHours_LtOverall = mfg.LtTotalUSEmploymentHours / 1000   '
        (in thousands)
239         Me.TotalUSEmploymentJobs_LtOverall = mfg.LtTotalUSEmploymentHours / Me.HoursPerYear
240         ' values by RC
241         For i = 0 To Me.nRC - 1
242             Me.TotalSales_ByRC(i) = mfg.RcTotalSales(i)
243             Me.AverageFuelEconomy_ByRC(i) = mfg.RcAverageFuelEconomy(i)
244             Me.AverageCurbWeight_ByRC(i) = mfg.RcAverageCurbWeight(i)
245             Me.AverageArea_ByRC(i) = mfg.RcAverageArea(i)
246             Me.AverageTechnologyCostsRPE_ByRC(i) = mfg.RcAverageTechCosts(i)
247             Me.AverageFinesIncurredRPE_ByRC(i) = mfg.RcAverageFinesOwed(i)
248             Me.AveragePriceIncrease_ByRC(i) = mfg.RcAveragePrice(i)
249             Me.TotalTechnologyCosts_ByRC(i) = mfg.RcTotalTechCosts(i) / 1000000 ' (in
        millions)
250             Me.TotalFinesOwed_ByRC(i) = mfg.RcFines(i) / 1000000                ' (in
        millions)
251             Me.TotalPriceIncrease_ByRC(i) = mfg.RcTotalPurchaseCosts(i) / 1000000 ' (in
        millions)
252             ReDim Me.ApplicationRatePerTechnology_ByRC(i)(nTech - 1)
253             ReDim Me.PenetrationRatePerTechnology_ByRC(i)(nTech - 1)
```

```
254             For j = 0 To nTech - 1
255                 Me.ApplicationRatePerTechnology_ByRC(i)(j) = mfg.
    RcApplicationRatePerTechnology(i)(j)
256                 Me.PenetrationRatePerTechnology_ByRC(i)(j) = mfg.
    RcPenetrationRatePerTechnology(i)(j)
257             Next j
258             Me.TotalUSEmploymentHours_ByRC(i) = mfg.RcTotalUSEmploymentHours(i) / 1000  '
    (in thousands)
259             Me.TotalUSEmploymentJobs_ByRC(i) = mfg.RcTotalUSEmploymentHours(i) / Me.
    HoursPerYear
260         Next i
261
262         ' resize UsedRCs and nUsedRCs arrays to include the current scenario
263         If IsNothing(UsedRCs) OrElse (Me.Scenario >= UsedRCs.Length) Then
264             ReDim Preserve nUsedLTRCs(Me.Scenario)
265             ReDim Preserve nUsedRCs(Me.Scenario)
266             ReDim Preserve UsedRCs(Me.Scenario)
267             ReDim UsedRCs(Me.Scenario)(nRC - 1)
268         End If
269
270         ' find the RC's that are used by mfg
271         For i = 0 To Me.nRC - 1
272             If mfg.RcCAFE(i) <> 0 AndAlso mfg.RcCAFE(i) <> Double.NaN Then
273                 ' if the CAFE level at RC-i is not 0 and not NaN,
274                 ' then this is a valid rc for the given mfg
275                 If UsedRCs(Me.Scenario)(i) <> True Then
276                     nUsedLTRCs(Me.Scenario) -= CInt(i <> 0)       ' do -= because CInt(true)
     = -1, instead of +1
277                     nUsedRCs(Me.Scenario) += 1
278                     UsedRCs(Me.Scenario)(i) = True
279                 End If
280             End If
281         Next i
282     End Sub
283
284     Private Sub CreateStackDelta(ByVal Scenario As Integer, _
285                                  ByVal MY As Integer, _
286                                  ByVal MfgIdx As Integer, _
287                                  ByVal OP As MathOP)
288         '
289         ' the 0-th scenario is the baseline scenario
290         Dim BASELINE As Integer = 0
291
292         Dim i As Integer
293         Dim BaselineMfg As ManufacturerStackClass
294         Dim ThisMfg As ManufacturerStackClass
295
296         ' get the baseline values for this mfg from the stack
297         For i = 0 To CAFE.MfgsStack(BASELINE)(MfgIdx).Size() - 1 Step 3
298             If CAFE.MfgsStack(BASELINE)(MfgIdx).Read(i).MY = MY Then
299                 BaselineMfg = CAFE.MfgsStack(BASELINE)(MfgIdx).Read(i)
300                 Exit For
301             End If
302         Next i
303         ' check to see if the baseline for mfg at MY exists
304         If IsNothing(BaselineMfg) Then
305             MsgBox("Cannot find a baseline scenario for mfg: " & CAFE.Mfgs(MfgIdx).MfgName
    & _
306                    " at model year: " & MY & ".", MsgBoxStyle.Exclamation, "Cannot find
    baseline")
307             Exit Sub
308         End If
309
310         ' get the values for the current mfg
311         ThisMfg = CAFE.MfgsStack(Me.Scenario)(MfgIdx).Peek()
312
313         Me.CreateStackDelta(Scenario, BaselineMfg, ThisMfg, OP)
314     End Sub
315
316     Protected Sub CreateStackDelta(ByVal Scenario As Integer, _
```

```
317                                     ByVal BaselineMfg As ManufacturerStackClass, _
318                                     ByVal ThisMfg As ManufacturerStackClass, _
319                                     ByVal OP As MathOP)
320         '
321         ' check if the current scenario is 0; init and exit if so
322         If Scenario = 0 Then
323             init(Double.Epsilon)
324             Return
325         End If
326
327         Dim i, j As Integer
328         Dim nTech As Integer
329
330         ' get the total number of technologies
331         nTech = TechnologiesClass.nTotalTechnologies
332
333         ' values overall
334         Me.TotalSales_Overall = Me.Math(ThisMfg.TotalSales_Overall, BaselineMfg.TotalSales_Overall, OP)
335         Me.AverageFuelEconomy_Overall = Me.Math(ThisMfg.AverageFuelEconomy_Overall, BaselineMfg.AverageFuelEconomy_Overall, OP)
336         Me.AverageCurbWeight_Overall = Me.Math(ThisMfg.AverageCurbWeight_Overall, BaselineMfg.AverageCurbWeight_Overall, OP)
337         Me.AverageArea_Overall = Me.Math(ThisMfg.AverageArea_Overall, BaselineMfg.AverageArea_Overall, OP)
338         Me.AverageTechnologyCostsRPE_Overall = Me.Math(ThisMfg.AverageTechnologyCostsRPE_Overall, BaselineMfg.AverageTechnologyCostsRPE_Overall, OP)
339         Me.AverageFinesIncurredRPE_Overall = Me.Math(ThisMfg.AverageFinesIncurredRPE_Overall, BaselineMfg.AverageFinesIncurredRPE_Overall, OP)
340         Me.AveragePriceIncrease_Overall = Me.Math(ThisMfg.AveragePriceIncrease_Overall, BaselineMfg.AveragePriceIncrease_Overall, OP)
341         Me.TotalTechnologyCosts_Overall = Me.Math(ThisMfg.TotalTechnologyCosts_Overall, BaselineMfg.TotalTechnologyCosts_Overall, OP)
342         Me.TotalFinesOwed_Overall = Me.Math(ThisMfg.TotalFinesOwed_Overall, BaselineMfg.TotalFinesOwed_Overall, OP)
343         Me.TotalPriceIncrease_Overall = Me.Math(ThisMfg.TotalPriceIncrease_Overall, BaselineMfg.TotalPriceIncrease_Overall, OP)
344         ReDim Me.ApplicationRatePerTechnology_Overall(nTech - 1)
345         ReDim Me.PenetrationRatePerTechnology_Overall(nTech - 1)
346         For i = 0 To nTech - 1
347             Me.ApplicationRatePerTechnology_Overall(i) = Me.Math(ThisMfg.ApplicationRatePerTechnology_Overall(i), BaselineMfg.ApplicationRatePerTechnology_Overall(i), OP)
348             Me.PenetrationRatePerTechnology_Overall(i) = Me.Math(ThisMfg.PenetrationRatePerTechnology_Overall(i), BaselineMfg.PenetrationRatePerTechnology_Overall(i), OP)
349         Next i
350         Me.TotalUSEmploymentHours_Overall = Me.Math(ThisMfg.TotalUSEmploymentHours_Overall, BaselineMfg.TotalUSEmploymentHours_Overall, OP)
351         Me.TotalUSEmploymentJobs_Overall = Me.Math(ThisMfg.TotalUSEmploymentJobs_Overall, BaselineMfg.TotalUSEmploymentJobs_Overall, OP)
352         ' values light truck overall
353         Me.TotalSales_LtOverall = Me.Math(ThisMfg.TotalSales_LtOverall, BaselineMfg.TotalSales_LtOverall, OP)
354         Me.AverageFuelEconomy_LtOverall = Me.Math(ThisMfg.AverageFuelEconomy_LtOverall, BaselineMfg.AverageFuelEconomy_LtOverall, OP)
355         Me.AverageCurbWeight_LtOverall = Me.Math(ThisMfg.AverageCurbWeight_LtOverall, BaselineMfg.AverageCurbWeight_LtOverall, OP)
356         Me.AverageArea_LtOverall = Me.Math(ThisMfg.AverageArea_LtOverall, BaselineMfg.AverageArea_LtOverall, OP)
357         Me.AverageTechnologyCostsRPE_LtOverall = Me.Math(ThisMfg.AverageTechnologyCostsRPE_LtOverall, BaselineMfg.AverageTechnologyCostsRPE_LtOverall, OP)
358         Me.AverageFinesIncurredRPE_LtOverall = Me.Math(ThisMfg.AverageFinesIncurredRPE_LtOverall, BaselineMfg.AverageFinesIncurredRPE_LtOverall, OP)
359         Me.AveragePriceIncrease_LtOverall = Me.Math(ThisMfg.AveragePriceIncrease_LtOverall, BaselineMfg.AveragePriceIncrease_LtOverall, OP)
360         Me.TotalTechnologyCosts_LtOverall = Me.Math(ThisMfg.TotalTechnologyCosts_LtOverall, BaselineMfg.TotalTechnologyCosts_LtOverall, OP)
361         Me.TotalFinesOwed_LtOverall = Me.Math(ThisMfg.TotalFinesOwed_LtOverall, BaselineMfg
```

```
.TotalFinesOwed_LtOverall, OP)
362         Me.TotalPriceIncrease_LtOverall = Me.Math(ThisMfg.TotalPriceIncrease_LtOverall, BaselineMfg.TotalPriceIncrease_LtOverall, OP)
363         ReDim Me.ApplicationRatePerTechnology_LtOverall(nTech - 1)
364         ReDim Me.PenetrationRatePerTechnology_LtOverall(nTech - 1)
365         For i = 0 To nTech - 1
366             Me.ApplicationRatePerTechnology_LtOverall(i) = Me.Math(ThisMfg.ApplicationRatePerTechnology_LtOverall(i), BaselineMfg.ApplicationRatePerTechnology_LtOverall(i), OP)
367             Me.PenetrationRatePerTechnology_LtOverall(i) = Me.Math(ThisMfg.PenetrationRatePerTechnology_LtOverall(i), BaselineMfg.PenetrationRatePerTechnology_LtOverall(i), OP)
368         Next i
369         Me.TotalUSEmploymentHours_LtOverall = Me.Math(ThisMfg.TotalUSEmploymentHours_LtOverall, BaselineMfg.TotalUSEmploymentHours_LtOverall, OP)
370         Me.TotalUSEmploymentJobs_LtOverall = Me.Math(ThisMfg.TotalUSEmploymentJobs_LtOverall, BaselineMfg.TotalUSEmploymentJobs_LtOverall, OP)
371         ' values by RC
372         For i = 0 To Me.nRC - 1
373             ' for all light truck RC's, check if the number of used RC's for this scenario
374             ' is NOT equal to the number of light truck RC's for the baseline scenario (Sn. 0)...
375             If (i >= 3) And (nUsedLTRCs(Scenario) <> nUsedLTRCs(0)) Then
376                 ' if number of RC's is NOT the same for this scenario and the baseline scenario,
377                 ' then do not calculate individual deltas for any light truck RC's, but instead use Epsilon
378                 Me.TotalSales_ByRC(i) = Double.Epsilon
379                 Me.AverageFuelEconomy_ByRC(i) = Double.Epsilon
380                 Me.AverageCurbWeight_ByRC(i) = Double.Epsilon
381                 Me.AverageArea_ByRC(i) = Double.Epsilon
382                 Me.AverageTechnologyCostsRPE_ByRC(i) = Double.Epsilon
383                 Me.AverageFinesIncurredRPE_ByRC(i) = Double.Epsilon
384                 Me.AveragePriceIncrease_ByRC(i) = Double.Epsilon
385                 Me.TotalTechnologyCosts_ByRC(i) = Double.Epsilon
386                 Me.TotalFinesOwed_ByRC(i) = Double.Epsilon
387                 Me.TotalPriceIncrease_ByRC(i) = Double.Epsilon
388                 ReDim Me.ApplicationRatePerTechnology_ByRC(i)(nTech - 1)
389                 ReDim Me.PenetrationRatePerTechnology_ByRC(i)(nTech - 1)
390                 For j = 0 To nTech - 1
391                     Me.ApplicationRatePerTechnology_ByRC(i)(j) = Double.Epsilon
392                     Me.PenetrationRatePerTechnology_ByRC(i)(j) = Double.Epsilon
393                 Next j
394                 Me.TotalUSEmploymentHours_ByRC(i) = Double.Epsilon
395                 Me.TotalUSEmploymentJobs_ByRC(i) = Double.Epsilon
396             Else
397                 ' if number of RC's is the same for this scenario and the baseline scenario,
398                 ' then calculate deltas for all light truck RC's
399                 '
400                 ' also calculate deltas for auto RC's
401                 Me.TotalSales_ByRC(i) = Me.Math(ThisMfg.TotalSales_ByRC(i), BaselineMfg.TotalSales_ByRC(i), OP)
402                 Me.AverageFuelEconomy_ByRC(i) = Me.Math(ThisMfg.AverageFuelEconomy_ByRC(i), BaselineMfg.AverageFuelEconomy_ByRC(i), OP)
403                 Me.AverageCurbWeight_ByRC(i) = Me.Math(ThisMfg.AverageCurbWeight_ByRC(i), BaselineMfg.AverageCurbWeight_ByRC(i), OP)
404                 Me.AverageArea_ByRC(i) = Me.Math(ThisMfg.AverageArea_ByRC(i), BaselineMfg.AverageArea_ByRC(i), OP)
405                 Me.AverageTechnologyCostsRPE_ByRC(i) = Me.Math(ThisMfg.AverageTechnologyCostsRPE_ByRC(i), BaselineMfg.AverageTechnologyCostsRPE_ByRC(i), OP)
406                 Me.AverageFinesIncurredRPE_ByRC(i) = Me.Math(ThisMfg.AverageFinesIncurredRPE_ByRC(i), BaselineMfg.AverageFinesIncurredRPE_ByRC(i), OP)
407                 Me.AveragePriceIncrease_ByRC(i) = Me.Math(ThisMfg.AveragePriceIncrease_ByRC(i), BaselineMfg.AveragePriceIncrease_ByRC(i), OP)
408                 Me.TotalTechnologyCosts_ByRC(i) = Me.Math(ThisMfg.TotalTechnologyCosts_ByRC(i), BaselineMfg.TotalTechnologyCosts_ByRC(i), OP)
409                 Me.TotalFinesOwed_ByRC(i) = Me.Math(ThisMfg.TotalFinesOwed_ByRC(i), BaselineMfg.TotalFinesOwed_ByRC(i), OP)
410                 Me.TotalPriceIncrease_ByRC(i) = Me.Math(ThisMfg.TotalPriceIncrease_ByRC(i),
```

```
     BaselineMfg.TotalPriceIncrease_ByRC(i), OP)
411                 ReDim Me.ApplicationRatePerTechnology_ByRC(i)(nTech - 1)
412                 ReDim Me.PenetrationRatePerTechnology_ByRC(i)(nTech - 1)
413                 For j = 0 To nTech - 1
414                     Me.ApplicationRatePerTechnology_ByRC(i)(j) = Me.Math(ThisMfg.
    ApplicationRatePerTechnology_ByRC(i)(j), BaselineMfg.ApplicationRatePerTechnology_ByRC
    (i)(j), OP)
415                     Me.PenetrationRatePerTechnology_ByRC(i)(j) = Me.Math(ThisMfg.
    PenetrationRatePerTechnology_ByRC(i)(j), BaselineMfg.PenetrationRatePerTechnology_ByRC
    (i)(j), OP)
416                 Next j
417                 Me.TotalUSEmploymentHours_ByRC(i) = Me.Math(ThisMfg.
    TotalUSEmploymentHours_ByRC(i), BaselineMfg.TotalUSEmploymentHours_ByRC(i), OP)
418                 Me.TotalUSEmploymentJobs_ByRC(i) = Me.Math(ThisMfg.
    TotalUSEmploymentJobs_ByRC(i), BaselineMfg.TotalUSEmploymentJobs_ByRC(i), OP)
419             End If
420         Next i
421     End Sub
422
423     Protected Sub CreateStackCommon()
424         Dim i, j As Integer
425         Dim nTech As Integer
426
427         nTech = TechnologiesClass.nTotalTechnologies     ' get number of techs
428
429         ' values total (overall and by RC)
430         ' first add by RC.part.
431         For i = 0 To Me.nRC - 1
432             Me.TotalSales(i) = Me.TotalSales_ByRC(i)
433             Me.AverageFuelEconomy(i) = Me.AverageFuelEconomy_ByRC(i)
434             Me.AverageCurbWeight(i) = Me.AverageCurbWeight_ByRC(i)
435             Me.AverageArea(i) = Me.AverageArea_ByRC(i)
436             Me.AverageTechnologyCostsRPE(i) = Me.AverageTechnologyCostsRPE_ByRC(i)
437             Me.AverageFinesIncurredRPE(i) = Me.AverageFinesIncurredRPE_ByRC(i)
438             Me.AveragePriceIncrease(i) = Me.AveragePriceIncrease_ByRC(i)
439             Me.TotalTechnologyCosts(i) = Me.TotalTechnologyCosts_ByRC(i)
440             Me.TotalFinesOwed(i) = Me.TotalFinesOwed_ByRC(i)
441             Me.TotalPriceIncrease(i) = Me.TotalPriceIncrease_ByRC(i)
442             ReDim Me.ApplicationRatePerTechnology(i)(nTech - 1)
443             ReDim Me.PenetrationRatePerTechnology(i)(nTech - 1)
444             For j = 0 To nTech - 1
445                 Me.ApplicationRatePerTechnology(i)(j) = Me.
    ApplicationRatePerTechnology_ByRC(i)(j)
446                 Me.PenetrationRatePerTechnology(i)(j) = Me.
    PenetrationRatePerTechnology_ByRC(i)(j)
447             Next j
448             Me.TotalUSEmploymentHours(i) = Me.TotalUSEmploymentHours_ByRC(i)
449             Me.TotalUSEmploymentJobs(i) = Me.TotalUSEmploymentJobs_ByRC(i)
450         Next i
451         ' then add light truck overall part
452         Me.TotalSales(nTotal - 2) = Me.TotalSales_LtOverall
453         Me.AverageFuelEconomy(nTotal - 2) = Me.AverageFuelEconomy_LtOverall
454         Me.AverageCurbWeight(nTotal - 2) = Me.AverageCurbWeight_LtOverall
455         Me.AverageArea(nTotal - 2) = Me.AverageArea_LtOverall
456         Me.AverageTechnologyCostsRPE(nTotal - 2) = Me.AverageTechnologyCostsRPE_LtOverall
457         Me.AverageFinesIncurredRPE(nTotal - 2) = Me.AverageFinesIncurredRPE_LtOverall
458         Me.AveragePriceIncrease(nTotal - 2) = Me.AveragePriceIncrease_LtOverall
459         Me.TotalTechnologyCosts(nTotal - 2) = Me.TotalTechnologyCosts_LtOverall
460         Me.TotalFinesOwed(nTotal - 2) = Me.TotalFinesOwed_LtOverall
461         Me.TotalPriceIncrease(nTotal - 2) = Me.TotalPriceIncrease_LtOverall
462         ReDim Me.ApplicationRatePerTechnology(nTotal - 2)(nTech - 1)
463         ReDim Me.PenetrationRatePerTechnology(nTotal - 2)(nTech - 1)
464         For i = 0 To nTech - 1
465             Me.ApplicationRatePerTechnology(nTotal - 2)(i) = Me.
    ApplicationRatePerTechnology_LtOverall(i)
466             Me.PenetrationRatePerTechnology(nTotal - 2)(i) = Me.
    PenetrationRatePerTechnology_LtOverall(i)
467         Next i
468         Me.TotalUSEmploymentHours(nTotal - 2) = Me.TotalUSEmploymentHours_LtOverall
469         Me.TotalUSEmploymentJobs(nTotal - 2) = Me.TotalUSEmploymentJobs_LtOverall
```

```
470         ' then add overall part
471         Me.TotalSales(nTotal - 1) = Me.TotalSales_Overall
472         Me.AverageFuelEconomy(nTotal - 1) = Me.AverageFuelEconomy_Overall
473         Me.AverageCurbWeight(nTotal - 1) = Me.AverageCurbWeight_Overall
474         Me.AverageArea(nTotal - 1) = Me.AverageArea_Overall
475         Me.AverageTechnologyCostsRPE(nTotal - 1) = Me.AverageTechnologyCostsRPE_Overall
476         Me.AverageFinesIncurredRPE(nTotal - 1) = Me.AverageFinesIncurredRPE_Overall
477         Me.AveragePriceIncrease(nTotal - 1) = Me.AveragePriceIncrease_Overall
478         Me.TotalTechnologyCosts(nTotal - 1) = Me.TotalTechnologyCosts_Overall
479         Me.TotalFinesOwed(nTotal - 1) = Me.TotalFinesOwed_Overall
480         Me.TotalPriceIncrease(nTotal - 1) = Me.TotalPriceIncrease_Overall
481         ReDim Me.ApplicationRatePerTechnology(nTotal - 1)(nTech - 1)
482         ReDim Me.PenetrationRatePerTechnology(nTotal - 1)(nTech - 1)
483         For i = 0 To nTech - 1
484             Me.ApplicationRatePerTechnology(nTotal - 1)(i) = Me.
    ApplicationRatePerTechnology_Overall(i)
485             Me.PenetrationRatePerTechnology(nTotal - 1)(i) = Me.
    PenetrationRatePerTechnology_Overall(i)
486         Next i
487         Me.TotalUSEmploymentHours(nTotal - 1) = Me.TotalUSEmploymentHours_Overall
488         Me.TotalUSEmploymentJobs(nTotal - 1) = Me.TotalUSEmploymentJobs_Overall
489     End Sub
490
491     Protected Sub RemoveNaNs()
492         ' convert all numbers with Double.NaN value to 0
493         Dim i, j As Integer
494         Dim nTech As Integer
495
496         nTech = TechnologiesClass.nTotalTechnologies        ' get number of techs
497
498         ' values overall
499         NanToZero(TotalSales_Overall)
500         NanToZero(AverageFuelEconomy_Overall)
501         NanToZero(AverageCurbWeight_Overall)
502         NanToZero(AverageArea_Overall)
503         NanToZero(AverageTechnologyCostsRPE_Overall)
504         NanToZero(AverageFinesIncurredRPE_Overall)
505         NanToZero(AveragePriceIncrease_Overall)
506         NanToZero(TotalTechnologyCosts_Overall)
507         NanToZero(TotalFinesOwed_Overall)
508         NanToZero(TotalPriceIncrease_Overall)
509         For i = 0 To nTech - 1
510             NanToZero(ApplicationRatePerTechnology_Overall(i))
511             NanToZero(PenetrationRatePerTechnology_Overall(i))
512         Next i
513         NanToZero(TotalUSEmploymentHours_Overall)
514         NanToZero(TotalUSEmploymentJobs_Overall)
515
516         ' values light truck overall
517         NanToZero(TotalSales_LtOverall)
518         NanToZero(AverageFuelEconomy_LtOverall)
519         NanToZero(AverageCurbWeight_LtOverall)
520         NanToZero(AverageArea_LtOverall)
521         NanToZero(AverageTechnologyCostsRPE_LtOverall)
522         NanToZero(AverageFinesIncurredRPE_LtOverall)
523         NanToZero(AveragePriceIncrease_LtOverall)
524         NanToZero(TotalTechnologyCosts_LtOverall)
525         NanToZero(TotalFinesOwed_LtOverall)
526         NanToZero(TotalPriceIncrease_LtOverall)
527         For i = 0 To nTech - 1
528             NanToZero(ApplicationRatePerTechnology_LtOverall(i))
529             NanToZero(PenetrationRatePerTechnology_LtOverall(i))
530         Next i
531         NanToZero(TotalUSEmploymentHours_LtOverall)
532         NanToZero(TotalUSEmploymentJobs_LtOverall)
533
534         ' values by RC
535         For i = 0 To Me.nTotal - 1
536             If i < Me.nRC Then
537                 NanToZero(TotalSales_ByRC(i))
```

```
538                 NanToZero(AverageFuelEconomy_ByRC(i))
539                 NanToZero(AverageCurbWeight_ByRC(i))
540                 NanToZero(AverageArea_ByRC(i))
541                 NanToZero(AverageTechnologyCostsRPE_ByRC(i))
542                 NanToZero(AverageFinesIncurredRPE_ByRC(i))
543                 NanToZero(AveragePriceIncrease_ByRC(i))
544                 NanToZero(TotalTechnologyCosts_ByRC(i))
545                 NanToZero(TotalFinesOwed_ByRC(i))
546                 NanToZero(TotalPriceIncrease_ByRC(i))
547                 For j = 0 To nTech - 1
548                     NanToZero(ApplicationRatePerTechnology_ByRC(i)(j))
549                     NanToZero(PenetrationRatePerTechnology_ByRC(i)(j))
550                 Next j
551                 NanToZero(TotalUSEmploymentHours_ByRC(i))
552                 NanToZero(TotalUSEmploymentJobs_ByRC(i))
553             End If
554
555             ' values overall and by RC
556             NanToZero(TotalSales(i))
557             NanToZero(AverageFuelEconomy(i))
558             NanToZero(AverageCurbWeight(i))
559             NanToZero(AverageArea(i))
560             NanToZero(AverageTechnologyCostsRPE(i))
561             NanToZero(AverageFinesIncurredRPE(i))
562             NanToZero(AveragePriceIncrease(i))
563             NanToZero(TotalTechnologyCosts(i))
564             NanToZero(TotalFinesOwed(i))
565             NanToZero(TotalPriceIncrease(i))
566             For j = 0 To nTech - 1
567                 NanToZero(ApplicationRatePerTechnology(i)(j))
568                 NanToZero(PenetrationRatePerTechnology(i)(j))
569             Next j
570             NanToZero(TotalUSEmploymentHours(i))
571             NanToZero(TotalUSEmploymentJobs(i))
572         Next i
573     End Sub
574
575     Public Sub Copy(ByRef dst As ManufacturerStackClass)
576         Copy(dst, Me)
577     End Sub
578
579     Public Shared Sub Copy(ByRef dst As ManufacturerStackClass, ByRef src As
        ManufacturerStackClass)
580         dst = New ManufacturerStackClass
581         CopyData(dst, src)
582     End Sub
583
584     Protected Shared Sub CopyData(ByRef dst As ManufacturerStackClass, ByRef src As
        ManufacturerStackClass)
585         Dim i As Integer
586
587         dst.Scenario = src.Scenario
588         dst.MY = src.MY
589         dst.Manufacturer = src.Manufacturer
590         ' values overall
591         dst.TotalSales_Overall = src.TotalSales_Overall
592         dst.AverageFuelEconomy_Overall = src.AverageFuelEconomy_Overall
593         dst.AverageCurbWeight_Overall = src.AverageCurbWeight_Overall
594         dst.AverageArea_Overall = src.AverageArea_Overall
595         dst.AverageTechnologyCostsRPE_Overall = src.AverageTechnologyCostsRPE_Overall
596         dst.AverageFinesIncurredRPE_Overall = src.AverageFinesIncurredRPE_Overall
597         dst.AveragePriceIncrease_Overall = src.AveragePriceIncrease_Overall
598         dst.TotalTechnologyCosts_Overall = src.TotalTechnologyCosts_Overall
599         dst.TotalFinesOwed_Overall = src.TotalFinesOwed_Overall
600         dst.TotalPriceIncrease_Overall = src.TotalPriceIncrease_Overall
601         dst.ApplicationRatePerTechnology_Overall = src.ApplicationRatePerTechnology_Overall
602         dst.PenetrationRatePerTechnology_Overall = src.PenetrationRatePerTechnology_Overall
603         dst.TotalUSEmploymentHours_Overall = src.TotalUSEmploymentHours_Overall
604         dst.TotalUSEmploymentJobs_Overall = src.TotalUSEmploymentJobs_Overall
605         ' values light truck overall
```

```
606         dst.TotalSales_LtOverall = src.TotalSales_LtOverall
607         dst.AverageFuelEconomy_LtOverall = src.AverageFuelEconomy_LtOverall
608         dst.AverageCurbWeight_LtOverall = src.AverageCurbWeight_LtOverall
609         dst.AverageArea_LtOverall = src.AverageArea_LtOverall
610         dst.AverageTechnologyCostsRPE_LtOverall = src.AverageTechnologyCostsRPE_LtOverall
611         dst.AverageFinesIncurredRPE_LtOverall = src.AverageFinesIncurredRPE_LtOverall
612         dst.AveragePriceIncrease_LtOverall = src.AveragePriceIncrease_LtOverall
613         dst.TotalTechnologyCosts_LtOverall = src.TotalTechnologyCosts_LtOverall
614         dst.TotalFinesOwed_LtOverall = src.TotalFinesOwed_LtOverall
615         dst.TotalPriceIncrease_LtOverall = src.TotalPriceIncrease_LtOverall
616         dst.ApplicationRatePerTechnology_LtOverall = src.
        ApplicationRatePerTechnology_LtOverall
617         dst.PenetrationRatePerTechnology_LtOverall = src.
        PenetrationRatePerTechnology_LtOverall
618         dst.TotalUSEmploymentHours_LtOverall = src.TotalUSEmploymentHours_LtOverall
619         dst.TotalUSEmploymentJobs_LtOverall = src.TotalUSEmploymentJobs_LtOverall
620         ' values by RC and total
621         For i = 0 To nTotal - 1
622             ' values by RC
623             If i < nRC Then
624                 dst.TotalSales_ByRC(i) = src.TotalSales_ByRC(i)
625                 dst.AverageFuelEconomy_ByRC(i) = src.AverageFuelEconomy_ByRC(i)
626                 dst.AverageCurbWeight_ByRC(i) = src.AverageCurbWeight_ByRC(i)
627                 dst.AverageArea_ByRC(i) = src.AverageArea_ByRC(i)
628                 dst.AverageTechnologyCostsRPE_ByRC(i) = src.AverageTechnologyCostsRPE_ByRC
        (i)
629                 dst.AverageFinesIncurredRPE_ByRC(i) = src.AverageFinesIncurredRPE_ByRC(i)
630                 dst.AveragePriceIncrease_ByRC(i) = src.AveragePriceIncrease_ByRC(i)
631                 dst.TotalTechnologyCosts_ByRC(i) = src.TotalTechnologyCosts_ByRC(i)
632                 dst.TotalFinesOwed_ByRC(i) = src.TotalFinesOwed_ByRC(i)
633                 dst.TotalPriceIncrease_ByRC(i) = src.TotalPriceIncrease_ByRC(i)
634                 dst.ApplicationRatePerTechnology_ByRC(i) = src.
        ApplicationRatePerTechnology_ByRC(i)
635                 dst.PenetrationRatePerTechnology_ByRC(i) = src.
        PenetrationRatePerTechnology_ByRC(i)
636                 dst.TotalUSEmploymentHours_ByRC(i) = src.TotalUSEmploymentHours_ByRC(i)
637                 dst.TotalUSEmploymentJobs_ByRC(i) = src.TotalUSEmploymentJobs_ByRC(i)
638             End If
639             ' values total (overall and by RC)
640             dst.TotalSales(i) = src.TotalSales(i)
641             dst.AverageFuelEconomy(i) = src.AverageFuelEconomy(i)
642             dst.AverageCurbWeight(i) = src.AverageCurbWeight(i)
643             dst.AverageArea(i) = src.AverageArea(i)
644             dst.AverageTechnologyCostsRPE(i) = src.AverageTechnologyCostsRPE(i)
645             dst.AverageFinesIncurredRPE(i) = src.AverageFinesIncurredRPE(i)
646             dst.AveragePriceIncrease(i) = src.AveragePriceIncrease(i)
647             dst.TotalTechnologyCosts(i) = src.TotalTechnologyCosts(i)
648             dst.TotalFinesOwed(i) = src.TotalFinesOwed(i)
649             dst.TotalPriceIncrease(i) = src.TotalPriceIncrease(i)
650             dst.ApplicationRatePerTechnology(i) = src.ApplicationRatePerTechnology(i)
651             dst.PenetrationRatePerTechnology(i) = src.PenetrationRatePerTechnology(i)
652             dst.TotalUSEmploymentHours(i) = src.TotalUSEmploymentHours(i)
653             dst.TotalUSEmploymentJobs(i) = src.TotalUSEmploymentJobs(i)
654         Next i
655     End Sub
656
657     Public Overloads Overrides Function ToArray() As Object()
658         Dim size As Integer = 40
659         Dim stack(size - 1) As Object
660
661         stack(0) = Me.Manufacturer
662         ' values overall
663         stack(1) = Me.TotalSales_Overall
664         stack(2) = Me.AverageFuelEconomy_Overall
665         stack(3) = Me.AverageCurbWeight_Overall
666         stack(4) = Me.AverageArea_Overall
667         stack(5) = Me.AverageTechnologyCostsRPE_Overall
668         stack(6) = Me.AverageFinesIncurredRPE_Overall
669         stack(7) = Me.AveragePriceIncrease_Overall
670         stack(8) = Me.TotalTechnologyCosts_Overall
```

```
671         stack(9) = Me.TotalFinesOwed_Overall
672         stack(10) = Me.TotalPriceIncrease_Overall
673         stack(11) = Me.ApplicationRatePerTechnology_Overall
674         stack(12) = Me.PenetrationRatePerTechnology_Overall
675         'stack(12) = Me.TotalUSEmploymentHours_Overall          ' employment hours should
        not be part of the stack
676         stack(13) = Me.TotalUSEmploymentJobs_Overall
677         ' values light truck overall
678         stack(14) = Me.TotalSales_LtOverall
679         stack(15) = Me.AverageFuelEconomy_LtOverall
680         stack(16) = Me.AverageCurbWeight_LtOverall
681         stack(17) = Me.AverageArea_LtOverall
682         stack(18) = Me.AverageTechnologyCostsRPE_LtOverall
683         stack(19) = Me.AverageFinesIncurredRPE_LtOverall
684         stack(20) = Me.AveragePriceIncrease_LtOverall
685         stack(21) = Me.TotalTechnologyCosts_LtOverall
686         stack(22) = Me.TotalFinesOwed_LtOverall
687         stack(23) = Me.TotalPriceIncrease_LtOverall
688         stack(24) = Me.ApplicationRatePerTechnology_LtOverall
689         stack(25) = Me.PenetrationRatePerTechnology_LtOverall
690         'stack(25) = Me.TotalUSEmploymentHours_ltOverall          ' employment hours should
         not be part of the stack
691         stack(26) = Me.TotalUSEmploymentJobs_LtOverall
692         ' values by RC
693         stack(27) = Me.TotalSales_ByRC
694         stack(28) = Me.AverageFuelEconomy_ByRC
695         stack(29) = Me.AverageCurbWeight_ByRC
696         stack(30) = Me.AverageArea_ByRC
697         stack(31) = Me.AverageTechnologyCostsRPE_ByRC
698         stack(32) = Me.AverageFinesIncurredRPE_ByRC
699         stack(33) = Me.AveragePriceIncrease_ByRC
700         stack(34) = Me.TotalTechnologyCosts_ByRC
701         stack(35) = Me.TotalFinesOwed_ByRC
702         stack(36) = Me.TotalPriceIncrease_ByRC
703         stack(37) = Me.ApplicationRatePerTechnology_ByRC
704         stack(38) = Me.PenetrationRatePerTechnology_ByRC
705         'stack(38) = Me.TotalUSEmploymentHours_ByRC          ' employment hours should
        not be part of the stack
706         stack(39) = Me.TotalUSEmploymentJobs_ByRC
707
708         Return stack
709     End Function
710
711     Public Overloads Function ToArray(ByVal GetTotal As Boolean) As Object()
712         If GetTotal Then
713             Dim size As Integer = 14
714             Dim stack(size - 1) As Object
715
716             stack(0) = Me.Manufacturer
717             ' values total (overall and by RC)
718             stack(1) = Me.TotalSales
719             stack(2) = Me.AverageFuelEconomy
720             stack(3) = Me.AverageCurbWeight
721             stack(4) = Me.AverageArea
722             stack(5) = Me.AverageTechnologyCostsRPE
723             stack(6) = Me.AverageFinesIncurredRPE
724             stack(7) = Me.AveragePriceIncrease
725             stack(8) = Me.TotalTechnologyCosts
726             stack(9) = Me.TotalFinesOwed
727             stack(10) = Me.TotalPriceIncrease
728             stack(11) = Me.ApplicationRatePerTechnology
729             stack(12) = Me.PenetrationRatePerTechnology
730             'stack(12) = Me.TotalUSEmploymentHours          ' employment hours should
        not be part of the stack
731             stack(13) = Me.TotalUSEmploymentJobs
732
733             Return stack
734         Else
735             Return Me.ToArray()
736         End If
```

```
737     End Function
738
739     Public Overloads Function ToArray(ByVal GetTotal As Boolean, ByVal GetUsedRCsOnly As
        Boolean) As Object()
740         Dim stack() As Object
741         Dim arr() As Object
742         'Dim stackIdx() As Object
743         Dim bidx As Integer
744         Dim eidx As Integer
745         Dim i, j As Integer
746         'Dim nRemoved As Integer = 0
747
748         If GetTotal Then
749             stack = Me.ToArray(True)
750             bidx = 1              ' the index of the first data array
751             eidx = 14             ' the index of the last data array
752         Else
753             stack = Me.ToArray()
754             bidx = 27             ' the index of the first data array_ByRC
755             eidx = 40             ' the index of the last data array_ByRC
756         End If
757
758         If GetUsedRCsOnly Then
759             For i = bidx To eidx - 1
760                 ReDim arr(stack(i).Length - 1)
761                 For j = 0 To arr.Length - 1
762                     arr(j) = stack(i)(j)
763                 Next j
764
765                 stack(i) = Me.GetUsedRCsOnly(arr)
766             Next i
767         End If
768
769         Return stack
770     End Function
771
772     Protected Function GetUsedRCsOnly(ByRef arr() As Object) As Object()
773         Dim i, j As Integer
774         Dim nRemoved As Integer = 0
775         Dim newArr() As Object
776
777         For i = 0 To nRC - 1
778             If Not Me.UsedRCs(Me.Scenario)(i) Then
779                 ' this rc is not used, set it to nothing
780                 arr(i) = Nothing
781                 nRemoved += 1
782             End If
783         Next i
784
785         ReDim newArr(arr.Length - nRemoved - 1)
786         j = 0
787         For i = 0 To arr.Length - 1
788             If Not IsNothing(arr(i)) Then
789                 newArr(j) = arr(i)
790                 j += 1
791             End If
792         Next i
793
794         Return newArr
795     End Function
796
797 End Class
798
799 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
800 ' This class stores data per model year
801 '''
802 Public Class ManufacturersStackClass
803     Inherits CafeStacksClass
804
805     Public Overrides Function Push(ByVal csc As CafeStackClass) As Boolean
```

```
806         ' create and copy the new ManufacturerStackClass
807         Dim entry As ManufacturerStackClass
808         ManufacturerStackClass.Copy(entry, csc)
809         '
810         ' add it to the stack
811         Me.MyStack(Me.MySize) = entry
812         Me.MySize += 1
813
814         If Me.MySize >= Me.MyStack.Length Then
815             ' array at max capacity; expand the array
816             ReDim Preserve Me.MyStack(Me.MySize + (Me.CapacityIncrement - 1))
817         End If
818
819         Return True
820     End Function
821
822 End Class
823
```

```
Public Class MarketSimulationClass
    '
    Public RecalcSalesCalibrationsDone As Boolean = False
    '
    Public Sub AllocateCosts(ByVal iSc As Integer, ByVal MY As Integer)
        CAFE.Profiling.FunctionStart()

        '
        ' This sub implements the cost allocation model for a single model year.
        ' It does so based on inputs that specify the cost allocation strategy assumed
        ' for each manufacturer.
        '
        ' Each vehicle is assigned a corresponding regulatory cost (RegCost).
        '
        ' Initialization
        Dim NumberOfManufacturers As Integer = CAFE.Mfgs.Length
        Dim NumberOfVehicles As Integer = CAFE.Vehicles.Length
        Dim Strategy() As Integer = New Integer(NumberOfManufacturers - 1) {}
        Dim Sales() As Double = New Double(NumberOfVehicles - 1) {} ' by vehicle
        Dim MSRP() As Double = New Double(NumberOfVehicles - 1) {} ' by vehicle
        Dim PriceElasticity() As Double = New Double(NumberOfVehicles - 1) {} ' by vehicle
        Dim TechCost() As Double = New Double(NumberOfVehicles - 1) {} ' by vehicle
        Dim Fines() As Double = New Double(NumberOfManufacturers - 1) {} ' by manufacturer
        Dim VehPseudoFine() As Double = New Double(NumberOfVehicles) {} ' used to allocate fines
        Dim MfgPseudoFine As Double = 0 ' used to allocate fines
        Dim RegCost() As Double = New Double(NumberOfVehicles - 1) {} ' by vehicle
        Dim TotalSales As Double = 0
        Dim TotalSalesRevenue As Double = 0
        Dim TotalTechCost As Double = 0
        Dim ElastInvSum As Double = 0
        Dim PctIncrease As Double = 0
        Dim i As Integer = 0 ' generic counter
        Dim iMfg As Integer = 0 ' manufacturer counter
        Dim iVeh As Integer = 0 ' vehicle counter
        Dim MfgNumVeh As Integer = 0 ' total number of vehicles for current manufacturer
        For iMfg = 0 To NumberOfManufacturers - 1
            Strategy(iMfg) = CAFE.Mfgs(iMfg).CostAllocationStrategy
            Fines(iMfg) = CAFE.Mfgs(iMfg).TotalFines
        Next
        For iVeh = 0 To NumberOfVehicles - 1
            Sales(iVeh) = CAFE.Vehicles(iVeh).Sales(MY)
            MSRP(iVeh) = CAFE.Vehicles(iVeh).MSRP(MY)
            PriceElasticity(iVeh) = GetPriceElasticity(iVeh, MY)
            TechCost(iVeh) = CAFE.Vehicles(iVeh).TechCost 'TO DO:  Add to Vehicles
        Next
        '
        ' Processing
        For iMfg = 0 To NumberOfManufacturers - 1
            MfgNumVeh = CAFE.Mfgs(iMfg).nValidVehicles
            TotalSales = 0
            TotalSalesRevenue = 0
            TotalTechCost = 0
            For i = 0 To MfgNumVeh - 1
                iVeh = CAFE.Mfgs(iMfg).ValidVehicleIndexes(i)
                If CAFE.Vehicles(iVeh).IsValid Then
                    TotalSales += Sales(iVeh)
                    TotalSalesRevenue += Sales(iVeh) * MSRP(iVeh)
                    TotalTechCost += Sales(iVeh) * TechCost(iVeh)
                End If
            Next

            ' TODO: Need to find a way to deal with RegCost for strategy-0 when TotalTechCost = 0 and Fines <> 0
            If Strategy(iMfg) = 0 AndAlso TotalTechCost = 0 AndAlso Fines(iMfg) > 0 Then
                ' if the cost allocation strategy used for this mfg is 0 and
                ' TotalTechCost is 0 while fines is not, set the strategy for
                ' this mfg to 1
                Strategy(iMfg) = 1
            End If
```

```
69
70              ' if the NMNL sheet is missing and the cost allocation strategy for this mfg is
    2,
71              ' set the cost allocation strategy to 0
72              If CAFE.NMNLIncluded AndAlso Strategy(iMfg) = 2 Then
73                  Strategy(iMfg) = 0
74              End If
75
76              Select Case Strategy(iMfg)
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111             Case Else
112                 ' First, allocate any technology costs on an as-incurred basis.
113                 For i = 0 To MfgNumVeh - 1
114                     iVeh = CAFE.Mfgs(iMfg).ValidVehicleIndexes(i)
115                     RegCost(iVeh) = TechCost(iVeh)
116                 Next
117                 ' Then allocate any fines.
118                 If Fines(iMfg) > 0 Then
119                     ' Calculate pseudofines for manufacturer and each vehicle model.
120                     MfgPseudoFine = 0
121                     ' ReDim VehPseudoFine(MfgNumVeh - 1)
122                     For iVeh = 0 To NumberOfVehicles - 1
123                         VehPseudoFine(iVeh) = 0
124                     Next
125                     For i = 0 To MfgNumVeh - 1
126                         iVeh = CAFE.Mfgs(iMfg).ValidVehicleIndexes(i)
127                         If CAFE.Vehicles(iVeh).IsValid Then
128                             VehPseudoFine(iVeh) = GetPseudoFine(iVeh, iSc, MY)
129                             MfgPseudoFine += VehPseudoFine(iVeh)
130                         End If
131                     Next
132                     ' Allocate fines based on share of total pseudofines.
133                     For i = 0 To MfgNumVeh - 1
134                         iVeh = CAFE.Mfgs(iMfg).ValidVehicleIndexes(i)
135                         If CAFE.Vehicles(iVeh).IsValid Then
136                             RegCost(iVeh) += Fines(iMfg) * (VehPseudoFine(iVeh) /
```

```
    MfgPseudoFine) / Sales(iVeh)
                            End If
                        Next
                    End If
            End Select
        Next
        '
        ' Termination
        For iVeh = 0 To NumberOfVehicles - 1
            CAFE.Vehicles(iVeh).RegCost = RegCost(iVeh)
        Next

        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
    End Sub ' AllocateCosts
    '
    Public Sub RecalcSales(ByVal MY As Integer, _
                           ByVal IncludeRegCost As Boolean, _
                           ByVal AdjustEstimates As Boolean)
        '
        CAFE.Profiling.FunctionStart()
```

```
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
244     End Sub ' RecalcSales
245     '
246     Public Sub ModelBaselineSales(ByVal MYList() As Integer)
247         ' suppress market share model if NMNL is missing
248         If CAFE.NMNLIncluded Then
249             '
250             ' This sub uses the market share model and the baseline MSRP values to develop a set
251             ' of modeled baseline sales estimates.  These are used to calibrate subsequent results
252             ' of the market share model.
253             '
254             ' This sub should be called only once, and should be called before any other calls
255             ' to RecalcSales.
256             '
257             ' Initialization
258             Dim nVeh As Integer = CAFE.Vehicles.Length
259             Dim i As Integer = 0 ' generic counter
260             Dim iVeh As Integer = 0 ' vehicle counter
261             Dim MY As Integer = 0
```

```
262             Dim MYIdx As Integer = 0
263             Dim TestSales As Double = 0
264
265             ' Processing
266             For i = 0 To MYList.Length - 1
267                 MY = MYList(i)
268                 MYIdx = CAFE.GMYI(MY)
269                 RecalcSales(MY, False, False)
270                 For iVeh = 0 To nVeh - 1
271                     CAFE.Vehicles(iVeh).ModeledBaselineSales(MYIdx) = 0
272                     TestSales = CAFE.Vehicles(iVeh).Sales(MYIdx)
273                     CAFE.Vehicles(iVeh).ModeledBaselineSales(MYIdx) = CAFE.Vehicles(iVeh).
    Sales(MYIdx)
274                     ' Return current sales volumes to original values
275                     CAFE.Vehicles(iVeh).Sales(MYIdx) = CAFE.MasterVehicles(iVeh).Sales
    (MYIdx)
276                 Next
277             Next
278             '
279             ' Termination
280             ' Set indicator regarding completion of calibration run.
281             RecalcSalesCalibrationsDone = True
282         End If
283     End Sub ' ModelBaselineSales
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
```

```
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404     End Function ' GetPseudoFine
405
406     Public Function GetPseudoFine(ByVal iVeh As Integer, ByVal iSc As Integer, ByVal MYIdx
        As Integer)
407
408         ' Initialization
409         Dim PseudoFine As Double = 0
410         Dim nVeh As Integer = CAFE.Vehicles.Length
411         Dim MfgIdx As Integer = CAFE.GetMfgIndex(CAFE.Vehicles(iVeh).Manufacturer)
412         Dim FE As Double = CAFE.Vehicles(iVeh).FuelEconomy
413         Dim Sales As Double = CAFE.Vehicles(iVeh).Sales(MYIdx)
414         Dim RegClass As Double = CAFE.Vehicles(iVeh).RegClass
415         Dim Standard As Double = CAFE.Standards.GetStandard(iSc, CAFE.GetMYearFromIndex
        (MYIdx), MfgIdx, RegClass)
416
417         ' Processing
418         PseudoFine = Math.Max(0, CAFE.Parameters.K_f * Sales * (Standard - FE))
419
420         ' Termination
421         Return PseudoFine
422
423     End Function ' GetPseudoFine
424
425 End Class
```

```
1  ' These imports can be revisited later, after the TempWriteActions class is eliminated.
2  Imports System
3  Imports System.IO
4  Imports System.Collections
5
6  Public Enum ComponentTypeEnum
7      General = 0
8      Engine = 1
9      Transmission = 2
10     Aero = 3
11 End Enum
12
13 Public Class ModellingLogic
14     Public Shared NoFinesWarningIssued As Boolean = False       ' so we only do the message box once per scenario (at most)
15     'Friend Compliance As ComplianceFinding
16     Friend Shared ModelErrorsFile As DiagModelErrorsFile
17     Private MyApproach As CAFE.ApproachEnum
18     Private MySecondBestOK As Boolean
19     Private NoFines As Boolean
20     Private TempLowestCost As Double = Double.MaxValue
21     Private FirstModelYear As Boolean                           ' to mark that we're in the lowest MY for this scenario with this data
22     Private StartModelingYear As Integer
23     Private CF_Stack() As CF_StackClass    ' For holding recent CFs for each mfg
24     Private CFReserved As ComplianceFinding     ' saves reserved compliances; may need to be changed into an array for multiple CF's
25
26     Public Sub New()
27         If DBUG Then
28             denter(Me, "New()")
29         End If
30     End Sub
31
32 #Region "Loops"
33
34     Public Sub Begin(Optional ByVal Iteration As Integer = 0)
35         CAFE.Profiling.FunctionStart()
36
37         If DBUG Then
38             denter(Me, "Begin()")
39         End If
40         'Dim OF As DiagModelRunFile = New DiagModelRunFile      ' Can't do this here until MasterMfg is populated
41         ModelErrorsFile = New DiagModelErrorsFile
42         Try
43             'CAFE.OperatingModes.StageAnalysis = CAFE.CAFE_Form.m_CxBoxStageAnalysis.Checked     ' Hack until we put it into Options form
44
45             Scenario_Loop(Iteration)
46         Catch ex As Exception
47             Dim ErrorText = "An exception was caught in ModelingLogic: " + Chr(13) + Chr(10)
48             ErrorText += "Reason: " + ex.Message + Chr(13) + Chr(10)
49             ErrorText += Chr(13) + Chr(10) + "Stack Trace: " + Chr(13) + Chr(10)
50             ErrorText += ex.StackTrace
51             MessageBox.Show(ErrorText, "ERROR IN CAFE MODEL", MessageBoxButtons.OK, MessageBoxIcon.Error)
52             ModelErrorsFile.WriteError(ex.Message)
53             ModelErrorsFile.WriteError(ErrorText)
54         End Try
55         'OF.Close()
56         ModelErrorsFile.Close()
57         ModelErrorsFile = Nothing
58
59         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
60     End Sub
61     Private Sub Scenario_Loop(ByVal Iteration)
62         CAFE.Profiling.FunctionStart()
63
```

```
64          If DBUG Then
65              denter(Me, "Scenario_Loop()")
66          End If
67
68          Dim iScenario As Integer
69          PreProcessScenario(Iteration)
70          Dim OF As DiagModelRunFile = New DiagModelRunFile    ' do this here because
        MasterMfg is populated in PreProcessScenario
71          For iScenario = 0 To CAFE.Scenarios.nScenarios - 1
72              CAFE.CurrentScenario = iScenario
73              ML_DebugWriteline("Starting Scenario " + iScenario.ToString)
74              Debug.Indent()
75              CAFE.CAFE_Form.SetStatusBarScenario(iScenario)
76              MY_Loop(iScenario)
77              If CAFE.ModelRunning = False Then
78                  Exit For
79              End If
80              CAFE.CAFE_Form.SetProgBarScenario(iScenario)
81              Debug.Unindent()
82              ML_DebugWriteline("Ending Scenario " + iScenario.ToString)
83          Next iScenario
84          OF.Close()
85          OF = Nothing
86          CAFE.CurrentScenario = -1
87          PostProcessScenario()
88
89          CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
90      End Sub
91
92      Private Sub MY_Loop(ByVal iSc As Integer)
93          CAFE.Profiling.FunctionStart()
94
95          If DBUG Then
96              denter(Me, "MY_Loop(iSc)")
97          End If
98
99          Dim iMY As Integer
100         Dim StartIdxMy As Integer
101         Dim StartMY As Integer
102         StartMY = PreProcessMY(iSc)                         ' Preprocess and get first valid
        MY
103         StartIdxMy = CAFE.GMYI(StartMY)
104         Me.StartModelingYear = StartMY
105         FirstModelYear = True
106
107         ' Preprocessing for market share model.
108         ' Make list of model years included.
109         Dim MYListLength As Integer = 0
110         Dim MYList() As Integer
111         For iMY = StartIdxMy To CAFE.ModelYearsSelected.Length - 1
112             If CAFE.ModelYearsSelected(iMY) Then
113                 MYListLength += 1
114                 ReDim Preserve MYList(MYListLength - 1)
115                 MYList(MYListLength - 1) = CAFE.GetMYearFromIndex(iMY)
116             End If
117         Next
118         ' Populate modeled baseline sales for calibration of market share model results.
119         If MYList.Length > 0 Then CAFE.MarketSimulation.ModelBaselineSales(MYList)
120
121         For iMY = StartIdxMy To CAFE.ModelYearsSelected.Length - 1
122             Dim MY As Integer = CAFE.GetMYearFromIndex(iMY)
123             CAFE.CurrentMY = MY
124             ML_DebugWriteLine("Starting MY " + MY.ToString)
125             Debug.Indent()
126             CAFE.CAFE_Form.SetStatusBarMY(MY)
127             If CAFE.ModelYearsSelected(iMY) Then
128                 ConvergenceLoop(iSc, MY)
129             End If
130             Application.DoEvents()
131             Application.DoEvents()
```

```
132             If CAFE.ModelRunning = False Then
133                 Exit For
134             End If
135             CAFE.CAFE_Form.SetProgBarMY(MY)
136             Debug.Unindent()
137             ML_DebugWriteLine("Ending MY " + MY.ToString)
138             FirstModelYear = False
139         Next iMY
140         CAFE.CurrentMY = -1
141         PostProcessMY(iSc)
142
143         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
144     End Sub
145
146     Private Sub ConvergenceLoop(ByVal iSc As Integer, ByVal MY As Integer)
147         CAFE.Profiling.FunctionStart()
148
149         Dim ConvergenceLoopNumber As Integer = 0
150         Dim iMfg As Integer = 0
151         Dim nMfg As Integer = CAFE.Mfgs.Length
152         Dim Rc As Integer = 0
153         Dim nRC As Integer = 0
154         Dim MfgTotalSales As Double = 0
155         Dim GreatestRcCAFEChange As Double = 0
156         Dim GreatestRcShareChange As Double = 0
157         Dim TerminateConvergenceLoop As Boolean = False
158         If DBUG Then
159             denter(Me, "ConvergenceLoop(iSc, MY)")
160         End If
161
162         If PreProcessConvergence(iSc, MY) Then
163             ' PreProcessConvergence returns TRUE if we're NOT doing stage-analysis mode.
164
165             ' Place this here for now... move to preprocess later
166             ReDim CF_Stack(nMfg - 1)
167             For i As Integer = 0 To nMfg - 1
168                 CF_Stack(i) = New CF_StackClass
169             Next
170             iMfg = 0
171
172             Do Until TerminateConvergenceLoop = True
173                 CAFE.CurrentConvergence = ConvergenceLoopNumber
174                 CAFE.CAFE_Form.SetStatusBarConv(ConvergenceLoopNumber)
175                 MFG_Loop(iSc, MY, ConvergenceLoopNumber)
176                 CAFE.MarketSimulation.AllocateCosts(iSc, CAFE.GMYI(MY))          ' Added
    for KG by JVS 23-aug-04
177
178                 '' compute totals and averages
179                 'For iMfg = 0 To nMfg - 1
180                 '    CAFE.Mfgs(iMfg).ComputeTotalsAndAverages(MY)
181                 'Next
182
183                 ' recalc sales
184                 CAFE.MarketSimulation.RecalcSales(MY, True, True)
185                 '
186                 ' Decide whether termination of ConvergenceLoop is allowed.
187                 If ConvergenceLoopNumber >= CAFE.NumberOfAllowedConvergenceLoops - 1 Then
188                     ' have reached maximum number of loops allowed.
189                     TerminateConvergenceLoop = True
190                 Else
191                     For iMfg = 0 To nMfg - 1
192                         nRC = CAFE.Mfgs(iMfg).nRC - 1
193                         ' For each manufacturer, calculate total sales for this
    convergence loop.
194                         MfgTotalSales = 0
195                         For Rc = 0 To nRC - 1
196                             MfgTotalSales += CAFE.Mfgs(iMfg).RcTotalSales(Rc)
197                         Next Rc
198                         ' Populate loop termination test values for this convergence loop.
199                         For Rc = 0 To nRC - 1
```

```
200                             CAFE.Mfgs(iMfg).ConvergenceLoopRcCAFE(Rc,
    ConvergenceLoopNumber) = CAFE.Mfgs(iMfg).RcCAFE(Rc)
201                             CAFE.Mfgs(iMfg).ConvergenceLoopRcShare(Rc,
    ConvergenceLoopNumber) = CAFE.Mfgs(iMfg).RcTotalSales(Rc) / MfgTotalSales
202                         Next Rc
203                     Next iMfg
204                     ' If this isn't the first loop, see if early termination is allowed.
205                     If ConvergenceLoopNumber > 0 Then
206                         GreatestRcCAFEChange = 0
207                         GreatestRcShareChange = 0
208                         For iMfg = 0 To nMfg - 1
209                             nRC = CAFE.Mfgs(iMfg).nRC - 1
210                             For Rc = 0 To nRC - 1
211                                 ' Find greatest change in RC-specific CAFE levels
212                                 If CAFE.Mfgs(iMfg).ConvergenceLoopRcCAFE(Rc,
    ConvergenceLoopNumber - 1) > 0 AndAlso
213                                 Math.Abs(CAFE.Mfgs(iMfg).ConvergenceLoopRcCAFE(Rc,
    ConvergenceLoopNumber) - _
214                                 CAFE.Mfgs(iMfg).ConvergenceLoopRcCAFE(Rc,
    ConvergenceLoopNumber - 1)) /
215                                 CAFE.Mfgs(iMfg).ConvergenceLoopRcCAFE(Rc,
    ConvergenceLoopNumber - 1) > _
216                                 GreatestRcCAFEChange Then GreatestRcCAFEChange = _
217                                 Math.Abs(CAFE.Mfgs(iMfg).ConvergenceLoopRcCAFE(Rc,
    ConvergenceLoopNumber) - _
218                                 CAFE.Mfgs(iMfg).ConvergenceLoopRcCAFE(Rc,
    ConvergenceLoopNumber - 1)) /
219                                 CAFE.Mfgs(iMfg).ConvergenceLoopRcCAFE(Rc,
    ConvergenceLoopNumber - 1)
220                                 ' Find greatest change in RC-specific share of each
    manufacturer's total sales.
221                                 If CAFE.Mfgs(iMfg).ConvergenceLoopRcShare(Rc,
    ConvergenceLoopNumber - 1) > 0 AndAlso _
222                                 Math.Abs(CAFE.Mfgs(iMfg).ConvergenceLoopRcShare(Rc,
    ConvergenceLoopNumber) - _
223                                 CAFE.Mfgs(iMfg).ConvergenceLoopRcShare(Rc,
    ConvergenceLoopNumber - 1)) /
224                                 CAFE.Mfgs(iMfg).ConvergenceLoopRcShare(Rc,
    ConvergenceLoopNumber - 1) > _
225                                 GreatestRcShareChange Then GreatestRcShareChange = _
226                                 Math.Abs(CAFE.Mfgs(iMfg).ConvergenceLoopRcShare(Rc,
    ConvergenceLoopNumber) - _
227                                 CAFE.Mfgs(iMfg).ConvergenceLoopRcShare(Rc,
    ConvergenceLoopNumber - 1)) /
228                                 CAFE.Mfgs(iMfg).ConvergenceLoopRcShare(Rc,
    ConvergenceLoopNumber - 1)
229                             Next Rc
230                         Next iMfg
231                         If GreatestRcCAFEChange < CAFE.AllowedRcCAFEChange AndAlso _
232                         GreatestRcShareChange < CAFE.AllowedRcShareChange Then
233                             TerminateConvergenceLoop = True
234                         End If
235                     End If
236                 End If
237                 ConvergenceLoopNumber += 1
238             Loop
239         End If
240
241         ' compute totals and averages
242         For iMfg = 0 To nMfg - 1
243             CAFE.Mfgs(iMfg).ComputeTotalsAndAverages(MY)
244         Next
245         Dim lWriteFinal As FinalTechTallyClass
246         For iMfg = 0 To CF_Stack.Length - 1
247             lWriteFinal = New FinalTechTallyClass(iSc, MY, iMfg, Me.CF_Stack(iMfg), Not
    FirstModelYear)
248         Next iMfg
249
250         CAFE.CurrentConvergence = -1
251         PostProcessConvergence(iSc, MY)
```

```
252
253         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
254     End Sub
255
256     Private Sub MFG_Loop(ByVal iSc As Integer, ByVal MY As Integer, ByVal iConv As
        Integer)
257         CAFE.Profiling.FunctionStart()
258
259         Dim NewTechCost As Double
260         Dim ExitLoop As Boolean = False
261         Dim ExitReason As DebugWriteActions.CLOSE_REASON = MyDEBUG.DebugWriteActions.
        CLOSE_REASON.Other
262         If DBUG Then
263             denter(Me, "MFG_Loop(iSc, MY)")
264         End If
265
266         Dim i, j As Integer
267         Dim iMY As Integer = CAFE.GMYI(MY)
268
269         PreProcessMfg(iSc, MY, iConv)
270         For i = 0 To CAFE.Mfgs.Length - 1
271             CAFE.CurrentMfg = i
272             ML_DebugWriteLine("Starting MFG(" + i.ToString + ") " + CAFE.Mfgs(i).MfgName)
273             Debug.Indent()
274             Dim CF As ComplianceFinding
275
276             ' initialize the current mfg
277             InitMfg(iSc, MY, iConv, i)
278
279             Dim TPcredits_last(CAFE.Mfgs(i).TradingPool.Length - 1) As Double
280             Dim TPcredits_now(CAFE.Mfgs(i).TradingPool.Length - 1) As Double
281             Dim OF As DebugWriteActions = New DebugWriteActions(iSc, MY, iConv, i)
282             Dim StackExhausted As Boolean = False
283
284             ' clear out CFReserved from the previous manufacturer
285             If CAFE.OperatingModes.ReserveTechsIfInCompliance Then
286                 Me.ResetReservedCF(i)
287             End If
288
289             Do
290                 Application.DoEvents()
291                 ExitLoop = False
292                 Dim OverComply As Boolean = False
293
294                 ' Examine credits visi-vi trading groups.
295                 ' If we've got something with credits which wasn't there
296                 ' when we ended the previous convergence loop, then we're going to peel-
        off
297                 If iConv > 0 And StackExhausted = False Then
298                     For tp As Integer = 0 To CAFE.Mfgs(i).TradingPool.Length - 1
299                         ' Calculate TP credits from start of MY
300                         ' We're going to stick with the arrays to make debug a bit easier
301                         TPcredits_last(tp) = _
302                             CAFE.Mfgs(i).SumCreditsInTradingPool(CAFE.Mfgs(i).TradingPool
        (tp), _
303                                 CAFE.Mfgs(i).RcCredit_Last_Converge, False)
304                         TPcredits_now(tp) = _
305                             CAFE.Mfgs(i).SumCreditsInTradingPool(CAFE.Mfgs(i).TradingPool
        (tp), _
306                                 CAFE.Mfgs(i).RcCredit, False)
307                         If (TPcredits_now(tp) > TPcredits_last(tp)) _
308                                 And (TPcredits_now(tp)) > 0 And (TPcredits_last(tp) > 0)
        Then
309                             'If OverComply = False Then
310                             '    Debug.WriteLine("Overcomply for Mfg(" + i.ToString + _
311                             '        ") [" + CAFE.Mfgs(i).MfgName + "] in scenario " + _
312                             '        iSc.ToString + " for converg " + iConv.ToString)
313                             'End If
314                             OverComply = True
315                         End If
```

```
316                     Next tp
317                 End If
318
319                 If ((OverComply = False) And (CAFE.Mfgs(i).TotalFines > 0)) Then
320                     ' Adding technology
321                     CF = Compliance_Loop(iSc, MY, iConv, i)
322
323                     If CAFE.OperatingModes.ReserveTechsIfInCompliance Then
324                         ' if this CF puts this MFG over compliance, reserve it, and try to find
325                         ' a "better/closer" match
326                         If Not IsNothing(CF) AndAlso CAFE.SumArray(CF.Fines_After) <= 0 Then
327                             ' reserve the last found CF
328                             Me.ReserveCF(MY, i, iConv, CF, OF)
329
330                             ' attemp to find a better CF match, that will still put this CF over compliance
331                             Do
332                                 ' get the next compliance finding
333                                 CF = Compliance_Loop(iSc, MY, iConv, i)
334
335                                 If Not IsNothing(CF) Then
336                                     ' check to see if the next CF puts this mfg in (or over) compliance
337                                     If CAFE.SumArray(CF.Fines_After) <= 0 Then
338                                         ' reserve the "in-compliance" CF
339                                         Me.ReserveCF(MY, i, iConv, CF, OF)
340                                     Else
341                                         Me.ReserveTechUsed(MY, i, CF)
342                                     End If
343                                 End If
344                             Loop Until CF Is Nothing
345
346                             ' at this point, we have looped through each possible CF
347                             ' and CFReserved has the best possible match
348                             CF = Me.CFReserved
349                         End If
350                     End If
351
352                     ' 4/13/04
353                     ' save new fe, new weight, new area, tech cost, and tech used
354                     ' for this mfg based on the most efficient ComplianceFinding
355                     If Not IsNothing(CF) Then
356                         NewTechCost = CAFE.Mfgs(i).TotalTechCosts + CF.TotalTechCost
357                         If (Me.NoFines = True
358                             Or NewTechCost <= CAFE.Mfgs(i).TotalFines) Then
359
360                             ' See if we've created an engine or transmission
361                             If CF.ComponentCreated Then
362                                 If CF.ComponentType = ComponentTypeEnum.Engine Then
363                                     CF.ComponentIdx = EnginesClass.Clone(CF.ComponentOldIdx, CF.VehIdx)
364                                 ElseIf CF.ComponentType = ComponentTypeEnum.Transmission Then
365                                     CF.ComponentIdx = TransmissionsClass.Clone(CF.ComponentOldIdx, CF.VehIdx)
366                                 End If
367                             End If
368
369                             OF.RecordAction(CF, iConv)                  ' Write a record of what's going on...
370
371                             ' get total techcost for this mfg
372                             CAFE.Mfgs(i).TotalTechCosts += CF.TotalTechCost
373                             ' then save individual values for each vehicle
374                             InstallTechnologies(iSc, MY, iConv, i, CF)
375
376                             CF_Stack(i).push(CF)    ' Save this compliance finding on the stack.
```

```
377
378                         Else
379                             'Come here if CF is nothing... nothing to do
380                         End If
381                     Else
382                         ' Update progress bars
383                         For pRC As Integer = 1 To CAFE.Mfgs(i).nRC - 1
384                             CAFE.CAFE_Form.SetCAFEProgress(pRC, CAFE.Mfgs(i).RcCAFE(pRC), CAFE.Mfgs(i).RcFines(pRC))
385                         Next pRC
386
387                         NewTechCost = CAFE.Mfgs(i).TotalTechCosts
388                     End If
389
390                     ' semi-post-process the mfg (recalc Credits/Fines/CAFE)
391                     DeInitMfg(iSc, MY, iConv, i)
392
393                     ' Look for escape...
394                     If (Not (CAFE.Mfgs(i).TotalFines > 0 And Me.NoFines)) Or _
395                         ((CAFE.Mfgs(i).TotalFines <= NewTechCost) And Not (Me.NoFines)) Then
396                         ExitLoop = True
397                         If CAFE.Mfgs(i).TotalFines <= 0 Then
398                             ExitReason = MyDEBUG.DebugWriteActions.CLOSE_REASON.FinesZero
399                         ElseIf CAFE.Mfgs(i).TotalFines <= NewTechCost Then
400                             ExitReason = MyDEBUG.DebugWriteActions.CLOSE_REASON.TechCost_GT_Fines
401                         Else
402                             ExitReason = MyDEBUG.DebugWriteActions.CLOSE_REASON.Other
403                         End If
404                     ElseIf (IsNothing(CF)) Then
405                         ExitLoop = True
406                         ExitReason = MyDEBUG.DebugWriteActions.CLOSE_REASON.ExhaustTech
407                     End If
408
409                 ElseIf OverComply Then
410                     ' Come here if we're in or over compliance.
411                     If CF_Stack(i).Count > 0 Then
412                         Dim MyCF As ComplianceFinding
413
414                         ' Examine TP settings.  Start with trucks (TP 3 and above),
415                         ' then cars. Try stripping technology an RC (or TP) at a time.
416                         If TPcredits_now.Length > 3 _
417                                 AndAlso ((TPcredits_now(3) > TPcredits_last(3) _
418                                     And TPcredits_now(3) > 0 And TPcredits_last(3) > 0)) Then
419                             MyCF = CF_Stack(i).PopByRC(3)
420                         ElseIf TPcredits_now.Length > 4 _
421                                 AndAlso ((TPcredits_now(4) > TPcredits_last(4) _
422                                     And TPcredits_now(4) > 0 And TPcredits_last(4) > 0)) Then
423                             MyCF = CF_Stack(i).PopByRC(4)
424                         ElseIf TPcredits_now.Length > 5 _
425                                 AndAlso ((TPcredits_now(5) > TPcredits_last(5) _
426                                     And TPcredits_now(5) > 0 And TPcredits_last(5) > 0)) Then
427                             MyCF = CF_Stack(i).PopByRC(5)
428                         ElseIf TPcredits_now.Length > 6 _
429                                 AndAlso ((TPcredits_now(6) > TPcredits_last(6) _
430                                     And TPcredits_now(6) > 0 And TPcredits_last(6) > 0)) Then
431                             MyCF = CF_Stack(i).PopByRC(6)
432                         ElseIf TPcredits_now.Length > 7 _
433                                 AndAlso ((TPcredits_now(7) > TPcredits_last(7) _
434                                     And TPcredits_now(7) > 0 And TPcredits_last(7) > 0)) Then
435                             MyCF = CF_Stack(i).PopByRC(7)
436                         ElseIf TPcredits_now.Length > 8 _
437                                 AndAlso ((TPcredits_now(8) > TPcredits_last(8) _
438                                     And TPcredits_now(8) > 0 And TPcredits_last(8) > 0)) Then
```

```
    Then
439                         MyCF = CF_Stack(i).PopByRC(8)
440                     ElseIf TPcredits_now.Length > 9 _
441                             AndAlso ((TPcredits_now(9) > TPcredits_last(9) _
442                                 And TPcredits_now(9) > 0 And TPcredits_last(9) > 0))
    Then
443                         MyCF = CF_Stack(i).PopByRC(9)
444                     ElseIf (TPcredits_now.Length > 4 And TPcredits_now.Length < 11) _
445                             AndAlso ((TPcredits_now(10) > TPcredits_last(10) _
446                                 And TPcredits_now(10) > 0 And TPcredits_last(10) > 0))
     Then
447                         MyCF = CF_Stack(i).PopByRC(10)
448                     ElseIf TPcredits_now.Length = 11 _
449                         AndAlso ((TPcredits_now(11) > 0 And TPcredits_last(11)) > 0)
    Then
450                         MyCF = CF_Stack(i).PopByRC(11)
451                     ElseIf TPcredits_now.Length > 1 _
452                             AndAlso ((TPcredits_now(1) > TPcredits_last(1) _
453                                 And TPcredits_now(1) > 0 And TPcredits_last(1) > 0))
    Then
454                         MyCF = CF_Stack(i).PopByRC(1)
455                     ElseIf TPcredits_now.Length > 2 _
456                             AndAlso ((TPcredits_now(2) > TPcredits_last(2) _
457                                 And TPcredits_now(2) > 0 And TPcredits_last(2) > 0))
    Then
458                         MyCF = CF_Stack(i).PopByRC(2)
459                     Else
460                         ' Can't find it through the prefered route (by RC),
461                         ' so just get it off the top of the stack.
462                         MyCF = CF_Stack(i).Pop()
463                     End If
464                     Dim Removed As Boolean = False
465                     If IsNothing(MyCF) = False Then
466                         Removed = Me.UnInstallTechnologies(iSc, MY, iConv, i, MyCF)
467                     Else
468                         Debug.WriteLine("mycf is null")
469                     End If
470                     If Removed Then
471                         ' update some data in the CF for reporting
472                         For rc As Integer = 0 To CAFE.Mfgs(i).RcFines.Length - 1
473                             MyCF.CAFE_Before(rc) = CAFE.Mfgs(i).RcCAFE(rc)
474                             MyCF.Fines_Before(rc) = CAFE.Mfgs(i).RcFines(rc)
475                         Next rc
476                         DeInitMfg(iSc, MY, iConv, i)
477                         ' update some more data in the CF for reporting
478                         For rc As Integer = 0 To CAFE.Mfgs(i).RcFines.Length - 1
479                             MyCF.CAFE_After(rc) = CAFE.Mfgs(i).RcCAFE(rc)
480                             MyCF.Fines_After(rc) = CAFE.Mfgs(i).RcFines(rc)
481                         Next rc
482                         For v As Integer = 0 To MyCF.VehIdx.Length - 1
483                             Dim dv As Double
484                             dv = MyCF.VehOrigFE(v)
485                             MyCF.VehOrigFE(v) = MyCF.VehNewFE(v)
486                             MyCF.VehNewFE(v) = dv
487                             dv = MyCF.VehOrigWeight(v)
488                             MyCF.VehOrigWeight(v) = MyCF.VehNewWeight(v)
489                             MyCF.VehNewWeight(v) = dv
490                         Next
491                         OF.RecordAction(MyCF, iConv, True)
492                         ExitLoop = False
493                     Else
494                         StackExhausted = True
495                     End If
496                 Else
497                     StackExhausted = True
498                 End If
499             Else
500                 ' Here if we're in compliance (and not overly over compliance...)
501                 ExitLoop = True
502                 If CF_Stack(i).Count = 0 Then
```

```
503                         ExitReason = MyDEBUG.DebugWriteActions.CLOSE_REASON.InCompliance
504                     Else
505                         ExitReason = MyDEBUG.DebugWriteActions.CLOSE_REASON.NoPeelOff
506                     End If
507                 End If
508             Loop While ExitLoop = False
509
510             ModellingLogic.ModelErrorsFile.EndMfg()
511
512             ' Close the diag output file
513             OF.Close(ExitReason, CAFE.Mfgs(i).TotalFines, NewTechCost)
514
515             Application.DoEvents()
516             Application.DoEvents()
517             If CAFE.ModelRunning = False Then
518                 Exit For
519             End If
520
521             CAFE.CAFE_Form.SetProgBarMfg(i)
522             Debug.Unindent()
523             ML_DebugWriteLine("Ending MFG(" + i.ToString + ") " + CAFE.Mfgs(i).MfgName)
524         Next i  ' for i = 0 to nMFG - 1
525         CAFE.CurrentMfg = -1
526         PostProcessMfg(iSc, MY, iConv)
527
528         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
529     End Sub
530
531
532     Private Function Compliance_Loop(ByVal iSc As Integer, ByVal MY As Integer, _
533                                      ByVal iConv As Integer, ByVal iMfg As Integer) As ComplianceFinding
534         '
535         CAFE.Profiling.FunctionStart()
536
537         '
538         If DBUG Then
539             denter(Me, "Compliance_Loop(iSc, MY, iMfg)")
540         End If
541
542         Dim lMfg As ManufacturerClass
543         Dim tidx As Integer
544         Dim eidx As Integer
545         Dim vidx As Integer
546         Dim Tests As TestInfo
547         Dim Imprv As Double
548         Dim Cost As Double
549         Dim Compliance As ComplianceFinding
550
551         Dim EscapeLoop As Boolean = False
552         PreProcessCompliance(iSc, MY, iConv, iMfg)
553
554         Do
555             If MyApproach = CAFE.ApproachEnum.LowCostFirst Then
556                 ' Exhaust low-cost-first technologies.
557                 Compliance = GetCompliance(iSc, MY, iConv, iMfg, True, MySecondBestOK)
558                 ' "Compliance is Nothing" means that the compliance finding does not exist <ms>
559                 If Compliance Is Nothing OrElse Compliance.IsEfficient = False Then
560                     ' reset approach for the remainder of this MFG in this MY/Scenario
561                     If CAFE.OperatingModes.AutoToggleApproach Then
562                         ML_DebugWriteline("Changing mode from LC to EC for in Sc: " + iSc.ToString _
563                                           + " MY: " + MY.ToString + " mfg: " + iMfg.ToString)
564                         MyApproach = CAFE.ApproachEnum.EconomicEfficiency
565                     Else
566                         MySecondBestOK = True
567                         If Compliance Is Nothing OrElse Compliance.SecondBest = False Then
568                             ' We're switching to second best.  Our last compliance was NOTHING,
```

```
569                             ' so let's take another stab at it with 2nd best = true
570                             Compliance = GetCompliance(iSc, MY, iConv, iMfg, True,
    MySecondBestOK)
571                         End If
572                     End If
573                 End If
574             End If
575             Application.DoEvents()
576             Application.DoEvents()
577             If CAFE.ModelRunning = False Then
578                 Exit Function
579             End If
580
581             If MyApproach = CAFE.ApproachEnum.EconomicEfficiency Then
582                 Compliance = GetCompliance(iSc, MY, iConv, iMfg, False, True)
583             End If
584             ' Set up escape clause (there is an escape clause but there's no such thing as
     a santa clause)
585             If (IsNothing(Compliance))
586             OrElse (Me.NoFines = False And Compliance.TotalTechCost <= CAFE.Mfgs(iMfg).
    TotalFines) _
587             OrElse (Me.NoFines And IsNothing(Compliance) = False) Then
588                 EscapeLoop = True
589             Else
590                 ' Not escaping ... if it's because tech costs are too big, set "Ignore" on
591                 ' these technology and go fish.
592                 If IsNothing(Compliance) = False _
593                 AndAlso (Me.NoFines = False And Compliance.TotalTechCost >= CAFE.Mfgs
    (iMfg).TotalFines) Then
594                     For t As Integer = 0 To Compliance.TechIdx.Length - 1
595                         Dim tx As Integer = Compliance.TechIdx(t)
596                         For vv As Integer = 0 To Compliance.VehIdx.Length - 1
597                             Dim vx As Integer = Compliance.VehIdx(vv)
598                             CAFE.Vehicles(vx).TechIgnored(tx) = True
599                         Next vv
600                     Next t
601                     Me.TempLowestCost = Math.Min(Compliance.TotalTechCost, Me.
    TempLowestCost)
602                 End If
603             End If
604         Loop While EscapeLoop = False
605         PostProcessCompliance(iSc, MY, iConv, iMfg)
606
607         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
608         Return Compliance
609     End Function
610 #End Region
611
612 #Region "Finding the best compliance"
613     ' Cycle through the different technology groups looking for the best
614     ' technology/vehicle application, if any...
615     Private Function GetCompliance(ByVal iSc As Integer, ByVal MY As Integer, _
616                                    ByVal iConv As Integer, ByVal iMfg As Integer, _
617                                    ByVal LC_First As Boolean, ByVal SecondBestOK As
    Boolean) As ComplianceFinding
618         '
619         CAFE.Profiling.FunctionStart()
620
621         '
622         If DBUG Then
623             denter(Me, "Compliance_LC_First(iSc, MY, iMfg)")
624         End If
625
626         Dim EngineFindings As ComplianceFinding
627         Dim TransmissionFindings As ComplianceFinding
628         Dim Findings(7) As ComplianceFinding
629
630         ' Engines first
631         Findings(0) = FindEffByTechGroup(iSc, MY, iConv, iMfg, TechnologiesClass.
    EngTechnologiesIdx, _
```

```
632                 CAFE.Mfgs(iMfg).EngineIndexes, ComponentTypeEnum.Engine, LC_First, SecondBestOK)
633
634         ' then transmissions
635         Findings(1) = FindEffByTechGroup(iSc, MY, iConv, iMfg, TechnologiesClass.TranTechnologiesIdx, _
636                 CAFE.Mfgs(iMfg).TransmissionIndexes, ComponentTypeEnum.Transmission, LC_First, SecondBestOK)
637
638         ' Accessory load
639         Findings(2) = FindEffByTechGroup(iSc, MY, iConv, iMfg, TechnologiesClass.AccLoadTechnologiesIdx, LC_First, SecondBestOK)
640
641         ' Dynamic load
642         Findings(3) = FindEffByTechGroup(iSc, MY, iConv, iMfg, TechnologiesClass.DLRTechnologiesIdx, LC_First, SecondBestOK)
643
644         ' Aerodynamic load reduction
645         Findings(4) = FindEffByTechGroup(iSc, MY, iConv, iMfg, TechnologiesClass.AeroTechnologiesIdx, _
646                 CAFE.Mfgs(iMfg).NamePlateIndexes, ComponentTypeEnum.Aero, LC_First, SecondBestOK)
647         ' Material Substitution Minus (Weight Reduction)
648         Findings(5) = FindEffByTechGroup(iSc, MY, iConv, iMfg, TechnologiesClass.MSMTechnologiesIdx, LC_First, SecondBestOK)
649
650         ' Material Substitution Plus (weight increase)
651         Findings(6) = FindEffByTechGroup(iSc, MY, iConv, iMfg, TechnologiesClass.MSPTechnologiesIdx, LC_First, SecondBestOK)
652
653         ' Other
654         Findings(7) = FindEffByTechGroup(iSc, MY, iConv, iMfg, TechnologiesClass.OtherTechnologiesIdx, LC_First, SecondBestOK)
655
656         ' Now select the best of the bunch.
657         Dim Finding As ComplianceFinding
658         Finding = ComplianceFinding.BestEff(Findings)
659
660         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
661         Return Finding
662     End Function
663
664     ' FindEffByTechGroup
665     ' This routine takes a technology group (Engines technology, transmissions...)
666     ' and a component list (engines, transmissions) and finds the impacts to the
667     ' vehicles associated with that component, looking for the best EFF.
668     ' Entry args:
669     '   iSc     Scenario number
670     '   iMY     index to model year
671     '   iConv   index to convergence loop
672     '   iMfg    index to the manufacturer
673     '   TechGroup   Array of technologies
674     '   ComponentList    ' The components (engines, transmissions)
675     '   ComponentType    ' 1 = engines, 2 = transmissions
676     '   LowCostFirst Boolean to indicate mode: Low-Cost-First or Economic Efficiency
677     '   SecondBestOK Boolean to indicate that it's ok to return 2nd best if 1st best not found
678     Private Function FindEffByTechGroup(ByVal iSc As Integer, _
679                                         ByVal MY As Integer, _
680                                         ByVal iConv As Integer, _
681                                         ByVal iMfg As Integer, _
682                                         ByVal TechGroup() As Integer, _
683                                         ByVal ComponentList() As Integer, _
684                                         ByVal ComponentType As ComponentTypeEnum, _
685                                         ByVal LowCostFirst As Boolean, _
686                                         ByVal SecondBestOK As Boolean) As ComplianceFinding
687         '
688         CAFE.Profiling.FunctionStart()
689
```

```
690         '
691         If DBUG Then
692             denter(Me, "LcFirstByTechGroup(iSc, MY, iMfg, TechGroup(), ComponentList(),
    ComponentType)")
693         End If
694
695         ' For each engine technology...
696         Dim lMfg As ManufacturerClass = CAFE.Mfgs(iMfg)
697         Dim tIdx As Integer ' Technology Index
698         Dim cIdx As Integer ' Component Index
699         Dim vIdx As Integer ' Vehicle Index
700
701         Dim Profiles() As TechProfile
702         Dim SkipIfEnableMisMatch As Boolean = CAFE.OperatingModes.SkipIfEnableMismatch
703         Dim SkipIfRCMisMatch As Boolean = CAFE.OperatingModes.SkipIfMultiRC
704         Dim SkipIfCarsAndTrucks As Boolean = CAFE.OperatingModes.SkipIfAutoLightTruckMix
705         Dim SplitIfEnableMisMatch As Boolean
706         Dim SplitIfRCMisMatch As Boolean
707         Dim SplitIfCarsAndTrucks As Boolean
708
709         Dim lVehList() As Integer
710
711         Dim Costx() As Double
712         Dim Imprvx() As Double
713         Dim BaseFEx() As Double
714         Dim NewFEx() As Double
715         Dim Areax() As Double
716         Dim Weightx() As Double
717         Dim DeltaCW() As Double
718         Dim Salesx() As Double
719
720         Dim Compliance As ComplianceFinding
721         Dim BestCompliance As ComplianceFinding = New ComplianceFinding
722         Dim SecondBestCompliance As ComplianceFinding = New ComplianceFinding
723
724         If Not IsNothing(TechGroup) Then            ' some tech groups may be empty <ms>
725             For t As Integer = 0 To TechGroup.Length - 1
726                 tIdx = TechGroup(t)
727                 For c As Integer = 0 To ComponentList.Length - 1
728
729                     Select Case ComponentType
730                         Case ComponentTypeEnum.General
731                             ' "Other" than engines or transmissions
732                             ReDim lVehList(0)
733                             lVehList(0) = ComponentList(c)      ' Set up a dummy vehicle
    list
734                         Case ComponentTypeEnum.Engine
735                             ' Engines
736                             cIdx = ComponentList(c)
737                             lVehList = CAFE.Engines(cIdx).ModelsList
738                             SplitIfEnableMisMatch = CAFE.OperatingModes.
    CreateEngineIfTechEnableMissMatch
739                             SplitIfRCMisMatch = CAFE.OperatingModes.
    CreateEngineIfMultipleRC
740                             SplitIfCarsAndTrucks = CAFE.OperatingModes.
    CreateEngineIfCarsInvolved
741
742                         Case ComponentTypeEnum.Transmission
743                             ' Transmissions
744                             cIdx = ComponentList(c)
745                             lVehList = CAFE.Transmissions(cIdx).ModelsList
746                             SplitIfEnableMisMatch = CAFE.OperatingModes.
    CreateTransmissionIfTechEnableMissMatch
747                             SplitIfRCMisMatch = CAFE.OperatingModes.
    CreateTransmissionIfMultipleRC
748                             SplitIfCarsAndTrucks = CAFE.OperatingModes.
    CreateTransmissionIfCarsInvolved
749                         Case ComponentTypeEnum.Aero
750                             ' Aerodynamic
751                             cIdx = ComponentList(c)
```

```
752                             lVehList = CAFE.Mfgs(iMfg).NamePlates(cIdx).ModelsList
753                             SplitIfEnableMisMatch = False
754                             SplitIfRCMisMatch = False
755                             SplitIfCarsAndTrucks = False
756                     End Select
757
758                     If IsNothing(lVehList) OrElse lVehList.Length = 0 _
759                         OrElse (ComponentType = ComponentTypeEnum.General _
760                             And CAFE.Vehicles(lVehList(0)).IsValid = False) Then
761                         ' No sense waiting around if nobody uses this component OR if vehicle is invalid
762                         GoTo NextC001
763                     End If
764
765                     Dim MultipleRC As Boolean = False
766                     Dim MixedEnables As Boolean = False
767                     Dim MixedAutos As Boolean = False
768
769                     Profiles = Nothing
770                     Dim ProfilesCollection As Collection = New Collection
771
772                     ' OK, PAY ATTENTION:
773                     ' We're going to use a collectionclass to hold the profiles.
774                     ' Initially, it will have the array of profiles we got from TouchTechnologies.
775                     ' Then, if there's more than one profile, we're going to examine the split
776                     ' options, and sub-divide the profiles -- and, potentially, make engines or transmissions.
777                     ' This, of course, will only happen if we're dealing with an engine or transmission technology.
778                     If TouchTechnologies(MY, iMfg, TechGroup, t, lVehList, Profiles) = False Then
779                         ' The technology is valid for at least one of the vehicles using this engine
780                         ' deal with it one 'profile' at a time
781                         If Profiles.Length > 1 Then
782                             ' There's more than one profile... get an idea of why
783                             TechProfile.CompareProfile(Profiles, MixedAutos, MultipleRC, MixedEnables)
784                         End If
785                         If (MultipleRC And SkipIfRCMisMatch) _
786                         Or (MixedEnables And SkipIfEnableMisMatch) _
787                         Or (MixedAutos And SkipIfCarsAndTrucks) Then
788                             ' our profiles are split and the options don't allow us to follow...
789                             GoTo NextC001
790                         End If
791                         ' Collect the profiles (yes, yes, in a collection class) based
792                         ' on operating mode qualifiers
793                         ProfilesCollection.Add(Profiles)        ' Add what we have for profiles...
794
795                         If (SplitIfRCMisMatch And MultipleRC) Then
796                             ' 1) Take the profile out of collection
797                             ' 2) Split it amoung RC lines
798                             ' 3) Fill the collection with the new profiles
799                             Dim TempProfiles() As TechProfile
800                             Dim ColCount = ProfilesCollection.Count
801                             For ColNum As Integer = 0 To colcount - 1
802                                 tempprofiles = CType(ProfilesCollection(1), TechProfile())
803                                 ProfilesCollection.Remove(1)                ' and then remove that item
804                                 Dim RCProfiles() As TechProfile
805                                 For ThisRC As Integer = 0 To CAFE.Mfgs(iMfg).nRC - 1
806                                     RCProfiles = TechProfile.GetRCProfiles(tempprofiles, ThisRC)
807                                     If Not IsNothing(RCProfiles) Then
808                                         ProfilesCollection.Add(RCProfiles)
809                                     End If
```

```
810                                 Next ThisRC
811                             Next ColNum
812                         ElseIf (MixedAutos And SplitIfCarsAndTrucks) Then
813                             ' 1) Take the profile out of collection
814                             ' 2) Split it along auto/truck lines (giving two arrays of profiles)
815                             ' 3) Fill the collection with these new profiles
816                             Dim TempProfiles() As TechProfile
817                             Dim ColCount = ProfilesCollection.Count
818                             For ColNum As Integer = 0 To ColCount - 1
819                                 tempprofiles = CType(ProfilesCollection(1), TechProfile())
820                                 ProfilesCollection.Remove(1)                ' and then remove that item
821                                 Dim AutoProfiles() As TechProfile           ' we'll split the item and add it back
822                                 Dim TruckProfiles() As TechProfile          ' into the collection
823                                 AutoProfiles = TechProfile.GetAutomobileProfiles(TempProfiles)
824                                 TruckProfiles = TechProfile.GetLightTruckProfiles(TempProfiles)
825                                 If Not IsNothing(AutoProfiles) Then
826                                     ProfilesCollection.Add(AutoProfiles)
827                                 End If
828                                 If Not IsNothing(TruckProfiles) Then
829                                     TruckProfiles(0).AddComponent(ComponentType)
830                                     ProfilesCollection.Add(TruckProfiles)
831                                 End If
832                             Next ColNum
833                             TempProfiles = CType(ProfilesCollection(1), TechProfile())
834                         End If
835                         If SplitIfEnableMisMatch And MixedEnables Then
836                             ' 1) Take the profile out of collection
837                             ' 2) Peal off any with different tech enable states...
838                             ' 3) Fill the collection with these new profiles
839                             Dim TempProfiles() As TechProfile
840                             Dim CollectCount = ProfilesCollection.Count
841                             For collectnum As Integer = 0 To CollectCount - 1
842                                 tempprofiles = CType(ProfilesCollection(1), TechProfile())
843                                 ProfilesCollection.Remove(1)                ' and then remove that item
844                                 Dim ProfilesA(-1) As TechProfile            ' we'll add them back when split along enable lines
845                                 Dim ProfilesB(-1) As TechProfile
846                                 If tempprofiles.Length > 1 Then
847                                     For ep As Integer = 0 To tempprofiles.Length - 1
848                                         If tempprofiles(ep).DiffEnab Then
849                                             Dim lB As Integer = ProfilesB.Length
850                                             ReDim ProfilesB(lB)
851                                             ProfilesB(lB) = tempprofiles(ep)
852                                         Else
853                                             Dim lA As Integer = ProfilesA.Length
854                                             ReDim Profiles(lA)
855                                             Profiles(lA) = tempprofiles(ep)
856                                         End If
857                                     Next ep
858                                     If Not ProfilesA.Length > 0 Then
859                                         ProfilesCollection.Add(ProfilesA)
860                                     End If
861                                     If Not ProfilesB.Length > 0 Then
862                                         ProfilesCollection.Add(ProfilesB)
863                                     End If
864                                 End If
865                             Next collectnum
866                         End If
867
868                         '
869                         ' Ok, we're set to work on things.
870                         ' All identified combinations of vehicle/technologies are packaged away as profiles
```

```
871                         ' in ProfilesCollections. Keep in mind that these are arrays of
    profiles.
872                         ' Pull out a set of profiles from the collection, and work on that
     group first
873                         Dim TheseProfiles() As TechProfile
874                         Dim FirstProfile As Boolean = True      ' first profile DOESN'T
    get a new engine or transm.
875
876                         For Each TheseProfiles In ProfilesCollection
877                             ' now work on each profile within that group.
878                             Compliance = New ComplianceFinding(iSc, MY, iConv, iMfg, Me.
    NoFines)
879                             For p As Integer = 0 To TheseProfiles.Length - 1
880
881                                 ' Get a set of arrays ready for the vehicle info
882                                 ReDim Costx(TheseProfiles(p).VehIdx.Length - 1)
883                                 ReDim Imprvx(TheseProfiles(p).VehIdx.Length - 1)
884                                 ReDim BaseFEx(TheseProfiles(p).VehIdx.Length - 1)
885                                 ReDim NewFEx(TheseProfiles(p).VehIdx.Length - 1)
886                                 ReDim Areax(TheseProfiles(p).VehIdx.Length - 1)
887                                 ReDim Weightx(TheseProfiles(p).VehIdx.Length - 1)
888                                 ReDim DeltaCW(TheseProfiles(p).VehIdx.Length - 1)
889                                 ReDim Salesx(TheseProfiles(p).VehIdx.Length - 1)
890                                 For v As Integer = 0 To TheseProfiles(p).VehIdx.Length - 1
891                                     vIdx = TheseProfiles(p).VehIdx(v)
892                                     BaseFEx(v) = CAFE.Vehicles(vIdx).FuelEconomy
893                                     Areax(v) = CAFE.Vehicles(vIdx).OverAllArea
894                                     Weightx(v) = CAFE.Vehicles(vIdx).CurbWeight
895                                     Salesx(v) = CAFE.Vehicles(vIdx).Sales(CAFE.GMYI(MY))
896                                 Next v
897
898                                 ' get impact settings for all the vehicles
899                                 GetTechImpact(MY, _
900                                               iMfg, _
901                                               TheseProfiles(p).TechIdx, _
902                                               TheseProfiles(p).VehIdx, _
903                                               BaseFEx, _
904                                               Costx, _
905                                               Imprvx, _
906                                               NewFEx, _
907                                               DeltaCW)
908
909                                 ' now we need to trim the list in case there are any holes
    ...
910                                 ' (for example, if one vehicle didn't have a weight
    setting)
911                                 For v As Integer = 0 To TheseProfiles(p).VehIdx.Length - 1
912                                     If (Costx(v) <> 0) And (BaseFEx(v) <> 0) And (NewFEx
    (v) <> 0) _
913                                         And (Areax(v) <> 0) And (Weightx(v) <> 0) Then
914                                         ' this one is a winner...
915
916                                         Compliance.AddVehicleInfo(TheseProfiles(p).VehIdx
    (v), _
917                                             TheseProfiles(p).TechIdx, Costx(v), Imprvx(v),
     BaseFEx(v), NewFEx(v), _
918                                             CAFE.Vehicles(TheseProfiles(p).VehIdx(v)).
    CurbWeight, _
919                                             DeltaCW(v), _
920                                             CAFE.Vehicles(TheseProfiles(p).VehIdx(v)).
    OverAllArea, _
921                                             Salesx(v))
922                                         Compliance.ComponentIdx = cIdx
923                                         Compliance.ComponentType = ComponentType
924                                         If FirstProfile = False Then
925                                             ' If it's other than the first profile, then
    we're adding engines or trans.
926                                             Compliance.ComponentCreated = True
927                                             Compliance.ComponentOldIdx = cIdx
928                                         End If
```

```
929                                 End If
930                             Next v
931
932                             If Not IsNothing(Compliance.VehIdx) Then
933                                 ' only calc fines if there are any vehicles
934                                 ' in this compliance instance
935                                 Dim ThisCAFE(-1) As Double
936                                 Dim ThisFine(-1) As Double
937
938                                 ' Calculate impacts on the MFG.
939                                 CAFE.Mfgs(iMfg).CalcImpacts(iSc, _
940                                                 MY, _
941                                                 Compliance.VehIdx, _
942                                                 Compliance.VehNewFE, _
943                                                 Compliance.VehArea, _
944                                                 Compliance.VehNewWeight, _
945                                                 ThisCAFE, _
946                                                 ThisFine)
947
948                                 Compliance.AddRCInfo(ThisCAFE, ThisFine)
949
950                                 ' Got it all. Now calculate EFF and compare if we've got the best
951                                 Compliance.CalcEff()
952                                 '                                                   Compliance.SetChoiceFound()
953                                 BestCompliance = ComplianceFinding.BestEff(BestCompliance, Compliance)
954                                 If LowCostFirst And _
955                                     (Not IsNothing(BestCompliance) AndAlso BestCompliance.IsEfficient = False) Then
956                                     BestCompliance = Nothing
957                                     SecondBestCompliance = ComplianceFinding.SecondBestEff(SecondBestCompliance, Compliance)
958                                 End If
959                                 'If LowCostFirst = False And _
960                                 '    (IsNothing(BestCompliance) = False AndAlso BestCompliance.ChoiceFound) = False Then
961                                 '    ' Not in low cost mode AND we don't have an efficient solution [yet].
962                                 '    ' Meanwhile, look for inefficient solutions
963                                 '    SecondBestCompliance = ComplianceFinding.SecondBestEff(SecondBestCompliance, _
964                                 '                                                           Compliance)
965                                 'End If
966                             End If
967                         Next p
968                         FirstProfile = False
969                     Next TheseProfiles
970                 End If  ' if techvalid
971 NextC001:           ' Because VB doesn't have a 'continue' statement
972             Next c ' component loop
973             If IsNothing(BestCompliance) = False And LowCostFirst Then
974                 If BestCompliance.ChoiceFound = True Then
975                     ' Got a low-cost winner. Let's blow this joint.
976                     Exit For    ' Exiting for t ...
977                 End If
978             End If
979         Next t ' engine-technology loop
980     End If
981     'If LowCostFirst = False And _
982     '    (IsNothing(BestCompliance) = False AndAlso BestCompliance.ChoiceFound = False) And _
983     '    SecondBestCompliance.SecondBest = True Then
984     '    ' Here if it's NOT low cost first AND we couldn't find an efficient solution
985     '    ' BUT DID find an inefficient solution.
986     '    Return SecondBestCompliance
987     'End If
988     If IsNothing(BestCompliance) = True And SecondBestOK Then
989         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
```

```
990             Return SecondBestCompliance
991         Else
992             CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
993             Return BestCompliance
994         End If
995     End Function
996
997     ' LcFirstByTechGroup (alternative entry w/o components)
998     ' This routine takes a technology group (Engines technology, vehicle accessories...)
999     ' and finds the impacts to the vehicles of the manufacturer, looking for the best EFF.
1000    ' Entry args:
1001    '   iSc     Scenario number
1002    '   iMY     index to model year
1003    '   iMfg    index to the manufacturer
1004    '   TechGroup   Array of technologies
1005    '   LowCostFirst Boolean to indicate mode: Low-Cost-First or Economic Efficiency
1006    '   SecondBestOK Boolean to indicate ok to return 2nd best if 1st best not found
1007    Private Function FindEffByTechGroup(ByVal iSc As Integer, _
1008                                        ByVal MY As Integer, _
1009                                        ByVal iConv As Integer, _
1010                                        ByVal iMfg As Integer, _
1011                                        ByVal TechGroup() As Integer, _
1012                                        ByVal LowCostFirst As Boolean, _
1013                                        ByVal SecondBestOK As Boolean) As ComplianceFinding
1014        If DBUG Then
1015            denter(Me, "LcFirstByTechGroup(iSc, MY, iMfg, TechGroup())")
1016        End If
1017
1018        Return FindEffByTechGroup(iSc, MY, iConv, iMfg, TechGroup, CAFE.Mfgs(iMfg).AllVehicleIndexes, _
1019                                  ComponentTypeEnum.General, LowCostFirst, SecondBestOK)
1020    End Function
1021 #End Region
1022
1023 #Region "Support/Calculation Routines"
1024     ' TouchTechnologies takes a list of technologies, an index to the current technology,
1025     ' and a list of vehicles.
1026     ' It scans the vehicles, checking the technologies and looking for sets of vehicles which
1027     ' exhibit the same profile. A profile is the technology enable pattern (taking
1028     ' technology-backfill into account), as well as Regulatory Class.
1029     '
1030     ' It will return an array of TestInfo structure, one for each differing profile.
1031     ' It will
1032     ' It returns FALSE if the tech is invalid on ALL vehicles on the list, true if valid on one or more
1033     Public Function TouchTechnologies(ByVal MY As Integer, _
1034                                       ByVal iMfg As Integer, _
1035                                       ByRef techList() As Integer, _
1036                                       ByVal t As Integer, _
1037                                       ByRef vehList() As Integer, _
1038                                       ByRef Profiles() As TechProfile) As Boolean
1039        If DBUG Then
1040            denter(Me, "TouchTechnologies(MY, iMfg, techList(), t, vehList(), TechValidOnAll, MultipleRC)")
1041        End If
1042
1043        ' Examine the technology for this vehicle -- check if valid -- do some paperwork
1044        ' The entry args:
1045        ' iMY -- Index to model year
1046        ' iMFG -- Index to Mfg
1047        ' techlist -- technology list (could be engines, transmissions, ...)
1048        ' t -- index into technology list
1049        ' vehList -- vehicles list (from an RC list or from engines or transmissions modellist)
1050        ' Profiles -- list of profiles of valid veh/technologies
1051
1052        Dim AllInvalid As Boolean = True
1053        'Dim AllValid As Boolean = True
```

```
1054        Dim tIdx As Integer
1055        Dim v As Integer
1056        Dim vIdx As Integer
1057
1058        Dim lProf As TechProfile
1059
1060        For v = 0 To vehList.Length - 1
1061            vIdx = vehList(v)
1062            lProf = New TechProfile
1063            If IsTechValid(MY, iMfg, techList(t), vIdx) Then
1064                ' Mark that we've found at least one valid veh/tech combo
1065                AllInvalid = False
1066
1067                ' Vehicle and current technology work. start a profile
1068                lProf.AddVehicle(vIdx)
1069                lProf.AddTechnology(techList(t))
1070
1071                ' If backfill .... check previous technologies
1072                If CAFE.OperatingModes.BackFillTechnology And t > 0 Then
1073                    For tt As Integer = t - 1 To 0 Step -1
1074                        If IsTechValid(MY, iMfg, techList(tt), vIdx) Then
1075                            ' backfill technology valid...add to list
1076                            lProf.AddTechnology(techList(tt))
1077                        End If
1078                    Next tt
1079                End If
1080                ' All finished adding to lProf. Now see if it matches any previous
    profiles
1081                If IsNothing(Profiles) = True OrElse Profiles.Length = 0 Then
1082                    ReDim Profiles(0)
1083                    Profiles(0) = lProf
1084                Else
1085                    Dim p As Integer
1086                    Dim m As Boolean = False
1087                    For p = 0 To Profiles.Length - 1
1088                        If TechProfile.CompareProfile(Profiles(p), lProf) Then
1089                            ' matches this profile...add the vehicle
1090                            Profiles(p).AddVehicle(vIdx)
1091                            m = True    ' m := match
1092                            Exit For
1093                        End If
1094                    Next p
1095                    If m = False Then
1096                        ' Culdn't find matching profile.  Add a new one
1097                        p = Profiles.Length
1098                        ReDim Preserve Profiles(p)
1099                        Profiles(p) = lProf
1100                    End If
1101                End If
1102            End If
1103        Next v
1104        Return AllInvalid
1105    End Function
1106
1107    ' Alternative entry when testing a single vehicle
1108    Public Function TouchTechnologies(ByVal MY As Integer, _
1109                                      ByVal iMfg As Integer, _
1110                                      ByRef techList() As Integer, _
1111                                      ByVal t As Integer, _
1112                                      ByRef vehIdx As Integer, _
1113                                      ByRef Profiles() As TechProfile) As Boolean
1114        If DBUG Then
1115            denter(Me, "TouchTechnologies(MY, iMfg, techList(), t, vehIdx, TechValidOnAll)
    ")
1116        End If
1117
1118        Dim vehlist(0) As Integer
1119        vehlist(0) = vehIdx
1120        Return TouchTechnologies(MY, iMfg, techList, t, vehlist, Profiles)
1121    End Function
```

```
1122
1123    ' GetTechImpact returns the cost and improvements and weight change
1124    Public Sub GetTechImpact(ByVal MY As Integer, _
1125                             ByVal iMfg As Integer, _
1126                             ByRef techList() As Integer, _
1127                             ByRef vehList() As Integer, _
1128                             ByRef BaseFE() As Double, _
1129                             ByRef Cost() As Double, _
1130                             ByRef Imprv() As Double, _
1131                             ByRef NewFE() As Double, _
1132                             ByRef DeltaCW() As Double)
1133        If DBUG Then
1134            denter(Me, "GetTechImpact(MY, iMfg, techList(), t, vehList(), BaseFE(), Cost() ↙
    , NewFE())")
1135        End If
1136
1137        ' Input args:
1138        ' iMY -- Index to model year
1139        ' iMfg -- Index to CAFE.Mfg
1140        ' techList -- list of technologies
1141        ' vehlist -- list of vehicles
1142        ' BaseFE -- list of baseline FEs for the vehicles in vehlist
1143        ' Cost -- array for returning the costs
1144        ' NewFE -- array for returning the FE improvements for the vehicles in vehlist
1145        ' DeltaCW -- change in Curb Weight -- if applicable.
1146
1147        ' Protect ourselves
1148        If (IsNothing(techList)) Or (IsNothing(vehList)) Or (IsNothing(Cost)) Then
1149            Exit Sub
1150        End If
1151        Dim tIdx As Integer
1152        Dim v As Integer
1153        Dim vIdx As Integer
1154        Dim t As Integer
1155
1156        ' initialize cost to 0 and handle curb weight init
1157        Dim OrigCW(vehList.Length - 1) As Double
1158        Dim UpdatedCW(vehList.Length - 1) As Double
1159        For v = 0 To vehList.Length - 1
1160            Cost(v) = 0
1161            Imprv(v) = 0
1162            DeltaCW(v) = 0
1163            OrigCW(v) = CAFE.Vehicles(vehList(v)).CurbWeight
1164            UpdatedCW(v) = OrigCW(v)
1165        Next v
1166
1167        For t = 0 To techList.Length - 1
1168            tIdx = techList(t)
1169            For v = 0 To vehList.Length - 1
1170                vIdx = vehList(v)
1171                If IsTechValid(MY, iMfg, tIdx, vIdx) Then
1172                    If CAFE.Technologies.IsMaterialSubstitution(CAFE.Vehicles(vIdx).. ↙
    NRC_ClassIdx, tIdx) Then
1173                        Dim kw As Double = CAFE.Technologies.GetKWeight(CAFE.Vehicles ↙
    (vIdx).NRC_ClassIdx, tIdx)
1174                        UpdatedCW(v) *= (1 - kw)
1175                        DeltaCW(v) = OrigCW(v) - UpdatedCW(v)
1176                        Cost(v) = CAFE.Technologies.GetCost(CAFE.Vehicles(vIdx).TechClass, ↙
     tIdx) * DeltaCW(v)
1177                    Else
1178                        Cost(v) += CAFE.Technologies.GetCost(CAFE.Vehicles(vIdx).TechClass ↙
    , tIdx)
1179                    End If
1180                    Imprv(v) += CAFE.Technologies.GetImprv(CAFE.Vehicles(vIdx).TechClass, ↙
    tIdx)
1181                End If
1182            Next v
1183        Next t
1184
1185        ' Now that all the improvements have been tallied, calculate the new FE
```

```
1186        For v = 0 To vehList.Length - 1
1187            NewFE(v) = BaseFE(v) / (1 - Imprv(v))
1188        Next v
1189    End Sub
1190
1191    Public Function IsTechValid(ByVal MY As Integer, _
1192                                ByVal iMfg As Integer, _
1193                                ByVal tIdx As Integer, _
1194                                ByVal vIdx As Integer, _
1195                                Optional ByVal IgnoreReserved As Boolean = False) As
    Boolean
1196        If DBUG Then
1197            denter(Me, "GetTechImpact(MY, iMfg, tIdx, vIdx)")
1198        End If
1199
1200        ' Return true if this technology is valid for this vehicle of this MFG in this
    model year
1201
1202        ' Currently MY has no say in the matter...
1203        Dim rcode As Boolean = False
1204        If CAFE.Vehicles(vIdx).IsValid Then
1205            ' interpret TechExcluded=Nothing as good <ms>
1206            rcode = ((IsNothing(CAFE.Mfgs(iMfg).TechExcluded) OrElse _
1207                      CAFE.Mfgs(iMfg).TechExcluded(tIdx) = False) And _
1208                    (CAFE.Vehicles(vIdx).TechIgnored(tIdx) = False) And _
1209                    (CAFE.Vehicles(vIdx).TechEnabled(tIdx) = True) And _
1210                    (CAFE.Vehicles(vIdx).TechUsed(tIdx) = False) And _
1211                    (IgnoreReserved Or CAFE.Vehicles(vIdx).TechReserved(tIdx) = False) And
    _
1212                    (CAFE.Mfgs(iMfg).TechExhausted(tIdx) = False))
1213
1214        End If
1215        If rcode = False And CAFE.CurrentConvergence = 1 Then
1216            Dim Xcode As Boolean
1217            Xcode = ((IsNothing(CAFE.Mfgs(iMfg).TechExcluded) OrElse _
1218                      CAFE.Mfgs(iMfg).TechExcluded(tIdx) = False) And _
1219                    (CAFE.Vehicles(vIdx).TechEnabled(tIdx) = True) And _
1220                    (CAFE.Vehicles(vIdx).TechUsed(tIdx) = False) And _
1221                    (CAFE.Vehicles(vIdx).TechReserved(tIdx) = False) And _
1222                    (CAFE.Mfgs(iMfg).TechExhausted(tIdx) = False))
1223            If Xcode Then
1224                Debug.Write("")
1225            End If
1226        End If
1227        Return rcode
1228    End Function
1229
1230
1231    ' Install technologies
1232    ' Install the technologies in the given Compliance Finding for the
1233    ' specified MFG/Convergence Loop/MY/Scenario
1234    Private Sub InstallTechnologies(ByVal iSC As Integer, ByVal MY As Integer, _
1235                                    ByVal iConv As Integer, ByVal iMfg As Integer, _
1236                                    ByVal CF As ComplianceFinding)
1237        '
1238        CAFE.Profiling.FunctionStart()
1239
1240        '
1241        Dim v As Integer
1242        Dim iMY As Integer = CAFE.GetMYearIndex(MY)
1243        Dim TechType As String
1244        Dim TechCost As Double = CF.TechCost(0)
1245        Dim TechImprv As Double = CF.TechImprv(0)
1246
1247        ' then save individual values for each vehicle
1248        For v = 0 To CF.VehIdx.Length - 1
1249            Dim vIdx As Integer = CF.VehIdx(v)
1250            Dim NewFE As Double = CF.VehNewFE(v)
1251            Dim NewWeight As Double = CF.VehNewWeight(v)
1252
```

```
1253            If NewWeight <> CAFE.Vehicles(vidx).CurbWeight Then
1254                If CAFE.Scenarios.Scenarios(iSC).LtRegClassAttribute.ToCharArray = "WEIGHT
    " Then
1255                    ' Weight changed AND the RCs are weight based,
1256                    ' so get new RC
1257                    Dim OldRC As Integer = CAFE.Vehicles(vidx).RegClass
1258                    Dim NewRC As Integer
1259                    NewRC = CAFE.Standards.GetRegClass(iSC, MY, vidx)
1260                    If NewRC <> OldRC Then
1261                        ' Swap RCs
1262                        CAFE.Mfgs(iMfg).RC_Move(OldRC, NewRC, vidx)
1263                        Dim OutText As String
1264                        OutText = "Veh Change RC. Old RC: " _
1265                                + OldRC.ToString _
1266                                + " NewRC: " _
1267                                + NewRC.ToString _
1268                                + "  "
1269                        ModellingLogic.ModelErrorsFile.WriteError(OutText)
1270                    End If
1271
1272                    ' Since we've got weight change on a weight based scenario.
1273                    ' we'd better recalc everything (STD, CAFE, Credits and Fines)
1274                    For iRc As Integer = 0 To CAFE.Mfgs(iMfg).nRC - 1
1275                        CAFE.Standards.GetStandard(iSC, MY, iMfg, iRc)
1276                    Next
1277                    CAFE.Mfgs(iMfg).CalcCAFE(MY, True)
1278                    CAFE.Mfgs(iMfg).CalcCredit(iSC, MY, True)
1279                    CAFE.Mfgs(iMfg).CalcFines(iSC, MY, True)
1280                End If
1281            End If
1282
1283            ' Loop through technologies for this vehicle
1284            ' If the tech is valid, set it as used.
1285            ' If the tech is Diesel, turn off other engine techs
1286            For t As Integer = 0 To CF.TechIdx.Length - 1
1287                Dim tIdx As Integer = CF.TechIdx(t)
1288                If IsTechValid(MY, iMfg, tidx, vIdx, True) Then
1289                    ' Mark the technology as used
1290                    CAFE.Vehicles(vidx).ApplyTechnology_(tidx, iMY)
1291                    If Array.IndexOf(TechnologiesClass.EngTechnologiesIdx, tidx) <> -1
    Then
1292                        CAFE.Engines(CAFE.Vehicles(vIdx).TrueEngineCode).TechUsed(tidx) =
    True
1293                    End If
1294                    If Array.IndexOf(TechnologiesClass.TranTechnologiesIdx, tidx) <> -1
    Then
1295                        CAFE.Transmissions(CAFE.Vehicles(vIdx).TrueTransmissionCode).
    TechUsed(tidx) = True
1296                    End If
1297                    If Array.IndexOf(TechnologiesClass.AeroTechnologiesIdx, tidx) <> -1
    Then
1298                        CAFE.Mfgs(CAFE.Vehicles(vIdx).MfgIdx).NamePlates(CAFE.Vehicles
    (vIdx).NamePlateIdx).TechUsed(tidx) = True
1299                    End If
1300                    Dim tClass As Integer = CAFE.Vehicles(vidx).NRC_ClassIdx
1301                    TechType = CAFE.Technologies.VehicleClasses(tClass).Technology(tidx).
    TechType.ToUpper
1302                    If CAFE.Technologies.IsDieselization(tClass, tidx) Then
1303                        ' Found dieselization being applied.
1304                        ' Go through all other engine technologies for this
1305                        ' veh and mark them as unused and unavailable
1306                        Dim EtIdx As Integer        ' Engine technology idx
1307                        For et As Integer = 0 To TechnologiesClass.EngTechnologiesIdx.
    Length - 1
1308                            EtIdx = TechnologiesClass.EngTechnologiesIdx(et)
1309                            If CAFE.Technologies.IsDieselization(tClass, EtIdx) = False
    Then
1310                                CAFE.Vehicles(vidx).RemoveTechnology_(EtIdx, iMY)
1311                                CAFE.Vehicles(vidx).TechApplies(EtIdx) = False
1312                            End If
```

```
1313                        Next et
1314                    End If
1315
1316                    ' "Install" the technology into the vehicle/engine/transmission's database
1317                    InstallTechnologyOnVehicle(vidx, tidx)
1318
1319                    ' Add it's usage to the MFG Applied count
1320                    CAFE.Mfgs(iMfg).TechApplied(tidx) += CAFE.Vehicles(vIdx).Sales(iMY)
1321                End If
1322            Next t
1323
1324            CAFE.Vehicles(vIdx).FuelEconomy = NewFE
1325            CAFE.Vehicles(vIdx).CurbWeight = NewWeight
1326            CAFE.Vehicles(vidx).TechCostByMY(CAFE.GMYI(MY)) += TechCost
1327
1328            If TechType <> "AERO" And TechType <> "ENGMOD" And TechType <> "TRMOD" Then
1329                CAFE.Vehicles(vIdx).TechCostVehTechnologies += TechCost
1330                CAFE.Vehicles(vidx).TechImprvVehTechnologies *= (1 - TechImprv)
1331            End If
1332        Next v
1333        ' Determine which component type to assign the cost to
1334        Dim engidx As Integer = CAFE.Vehicles(CF.VehIdx(0)).TrueEngineCode
1335        If TechType = "AERO" Then
1336            Dim mfgidx As Integer = CAFE.Vehicles(CF.VehIdx(0)).MfgIdx
1337            Dim npidx As Integer = CAFE.Vehicles(CF.VehIdx(0)).NamePlateIdx
1338            CAFE.Mfgs(mfgidx).NamePlates(npidx).TechCost += TechCost
1339            CAFE.Mfgs(mfgidx).NamePlates(npidx).TechImprv *= (1 - TechImprv)
1340        ElseIf TechType = "ENGMOD" Then
1341            CAFE.Engines(engidx).TechCost += TechCost
1342            CAFE.Engines(engidx).TechImprv *= (1 - TechImprv)
1343        ElseIf TechType = "TRMOD" Then
1344            Dim tranidx As Integer = CAFE.Vehicles(CF.VehIdx(0)).TrueTransmissionCode
1345            CAFE.Transmissions(tranidx).TechCost += TechCost
1346            CAFE.Transmissions(tranidx).TechImprv *= (1 - TechImprv)
1347        End If
1348        CF.TechType = TechType  ' save the type in case we need to uninstall the tech
1349
1350        ' Now that that's done with, we'll loop through VEH and TECH again for phase-in
1351        For v = 0 To CF.VehIdx.Length - 1
1352            Dim vIdx As Integer = CF.VehIdx(v)
1353            Dim NrcClass As Integer = CAFE.Vehicles(vidx).NRC_ClassIdx
1354            For t As Integer = 0 To CF.TechIdx.Length - 1
1355                Dim tIdx As Integer = CF.TechIdx(t)
1356                If CAFE.Mfgs(iMfg).TechExhausted(tidx) = False Then
1357                    Dim Applied As Double = CAFE.Mfgs(iMfg).TechApplied(tidx)
1358                    Dim MY_TechIntroduced As Integer =
1359                        CAFE.Technologies.VehicleClasses(NrcClass).Technology(tidx).YearAvailable
1360                    Dim PhaseIn As Double =
1361                        CAFE.Technologies.VehicleClasses(NrcClass).Technology(tidx).PhaseIn
1362                    PhaseIn = Math.Min(1, PhaseIn * (MY - Math.Max(MY_TechIntroduced, Me.StartModelingYear) + 1))
1363                    Dim tSales As Double = CAFE.Mfgs(iMfg).TotalSales
1364                    If (Applied + 1) >= (tSales * PhaseIn) Then
1365                        CAFE.Mfgs(iMfg).TechExhausted(tidx) = True
1366                        ModellingLogic.ModelErrorsFile.WriteError("Exhausting Tech: " + tidx.ToString + _
1367                            "(" + CAFE.Technologies.VehicleClasses(NrcClass).Technology(tidx).Abbr + ")")
1368                        ModellingLogic.ModelErrorsFile.WriteString("vidx: " + vidx.ToString)
1369                        ModellingLogic.ModelErrorsFile.WriteString("Applied:," + Applied.ToString)
1370                        ModellingLogic.ModelErrorsFile.WriteString("TSales:," + tSales.ToString)
1371                        ModellingLogic.ModelErrorsFile.WriteString("Phasein:," + PhaseIn.ToString)
1372                    End If
```

```
1373                End If
1374            Next t
1375        Next v
1376
1377        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1378    End Sub
1379
1380    ' Take the given technology being applied to the given vehicle
1381    ' and change the vehicle's attributes (or it's engine's or transmission's)
1382    ' to reflect the new technology.
1383    Public Sub InstallTechnologyOnVehicle(ByVal vIdx As Integer, ByVal tIdx As Integer)
1384        Dim EngIdx As Integer = CAFE.Vehicles(vIdx).TrueEngineCode
1385        Dim TranIdx As Integer = CAFE.Vehicles(vIdx).TrueTransmissionCode
1386
1387        Select Case tIdx
1388            Case TechnologiesClass.TechIndexes.tEngLUB
1389                CAFE.Engines(EngIdx).EngineOilViscosity = "LUB1"
1390            Case TechnologiesClass.TechIndexes.tEngOHC
1391                CAFE.Engines(EngIdx).ValvesCylinder = 4
1392            Case TechnologiesClass.TechIndexes.tTrn5SP
1393                CAFE.Transmissions(TranIdx).Control = "A"
1394                CAFE.Transmissions(TranIdx).NumberForwardGears = 5
1395            Case TechnologiesClass.TechIndexes.tTrn6SP
1396                CAFE.Transmissions(TranIdx).Control = "A"
1397                CAFE.Transmissions(TranIdx).NumberForwardGears = 6
1398            Case TechnologiesClass.TechIndexes.tTrnASL
1399                CAFE.Transmissions(TranIdx).Logic = "A"
1400            Case TechnologiesClass.TechIndexes.tTrnCVT
1401                CAFE.Transmissions(TranIdx).Type = "CVT"
1402                CAFE.Transmissions(TranIdx).NumberForwardGears = 0
1403                CAFE.Transmissions(TranIdx).Logic = ""
1404            Case TechnologiesClass.TechIndexes.tTrnAST
1405                CAFE.Transmissions(TranIdx).Type = "C"
1406                CAFE.Transmissions(TranIdx).NumberForwardGears = 6
1407                CAFE.Transmissions(TranIdx).Control = "A"
1408            Case TechnologiesClass.TechIndexes.tEngVVLT
1409                CAFE.Engines(EngIdx).ValveLift = "SV"
1410            Case TechnologiesClass.TechIndexes.tEngSUP
1411                CAFE.Engines(EngIdx).Aspiration = "S"
1412            Case TechnologiesClass.TechIndexes.tEngCVA
1413                CAFE.Engines(EngIdx).ValvetrainDesign = "C"
1414            Case TechnologiesClass.TechIndexes.tEngIVT
1415                CAFE.Engines(EngIdx).ValveLift = "CV"
1416            Case TechnologiesClass.TechIndexes.tTrnACV
1417                CAFE.Transmissions(TranIdx).Type = "ACVT"
1418                CAFE.Transmissions(TranIdx).NumberForwardGears = 0
1419            Case TechnologiesClass.TechIndexes.tEngDSL
1420                With CAFE.Engines(EngIdx)
1421                    .ValveActuationTiming = "F"
1422                    .ValveLift = "F"
1423                    .Fuel = "D"
1424                    .FuelSystem = "DI"
1425                    .Aspiration = "T"
1426                    .AirFuelRatio = 0
1427                    .MinCompressionRatio = 0
1428                    .MaxCompressionRatio = 0
1429                    .Cycle = "D"
1430                End With
1431                CAFE.Vehicles(vIdx).MOBILE6Class = CAFE.Vehicles(vIdx).AssignMobile6Class
1432            Case TechnologiesClass.TechIndexes.tTrnHEV
1433                CAFE.Vehicles(vIdx).HybridType = "E"
1434
1435                ' For material substitution ... if we install MS1, then we're allowed to
    enable MS2
1436                ' And... if we install MS2, then we're allowed to install MS3
1437            Case TechnologiesClass.TechIndexes.tMS_MS1
1438                CAFE.Vehicles(vIdx).TechEnabled(tIdx) = CAFE.Vehicles(vIdx).
    TechMS2Enablable
1439            Case TechnologiesClass.TechIndexes.tMS_MS2
1440                CAFE.Vehicles(vIdx).TechEnabled(tIdx) = CAFE.Vehicles(vIdx).
```

```
    TechMS3Enablable
1441        End Select
1442    End Sub
1443
1444    '''''''''''''''''''''
1445    ' UnInstallTechnologies
1446    ' For convergence loops > 0, peel off technology and recalc CAFE/STD/Fine for
1447    ' the MFG. Be sure to reset vehicle attributes accordingly
1448    '
1449    ' Returns TRUE if a tech was repealed, FALSE if not
1450    Private Function UnInstallTechnologies(ByVal iSc As Integer, ByVal MY As Integer, _
1451                                           ByVal iConv As Integer, ByVal iMfg As Integer, ↙
    _
1452                                           ByVal CF As ComplianceFinding) As Boolean
1453        '
1454        CAFE.Profiling.FunctionStart()
1455
1456        '
1457        Dim v As Integer
1458        Dim vIdx As Integer
1459        Dim eIdx As Integer
1460        Dim tIdx As Integer
1461
1462        If CF.Convergence = iConv Then
1463            ' Can't repeal technology applied in the same convergence loop.
1464            CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1465            Return False
1466        End If
1467
1468        ' Peel off technology costs for Nameplate, Engine or Transmission.
1469        If CF.TechType = "AERO" Then
1470            Dim mfgidx As Integer = CAFE.Vehicles(CF.VehIdx(0)).MfgIdx
1471            Dim npidx As Integer = CAFE.Vehicles(CF.VehIdx(0)).NamePlateIdx
1472            CAFE.Mfgs(mfgidx).NamePlates(npidx).TechCost -= CF.TechCost(0)
1473            CAFE.Mfgs(mfgidx).NamePlates(npidx).TechImprv -= CF.TechImprv(0)
1474        ElseIf CF.TechType = "ENGMOD" Then
1475            Dim engidx As Integer = CAFE.Vehicles(CF.VehIdx(0)).TrueEngineCode
1476            CAFE.Engines(engidx).TechCost -= CF.TechCost(0)
1477            CAFE.Engines(engidx).TechImprv -= CF.TechImprv(0)
1478        ElseIf CF.TechType = "TRMOD" Then
1479            Dim tranidx As Integer = CAFE.Vehicles(CF.VehIdx(0)).TrueTransmissionCode
1480            CAFE.Transmissions(tranidx).TechCost -= CF.TechCost(0)
1481            CAFE.Transmissions(tranidx).TechImprv -= CF.TechImprv(0)
1482        End If
1483
1484        For v = 0 To CF.VehIdx.Length - 1
1485            vIdx = CF.VehIdx(v)
1486            eIdx = CAFE.Vehicles(vIdx).TrueEngineCode
1487            tIdx = CAFE.Vehicles(vIdx).TrueTransmissionCode
1488            With CAFE.Vehicles(vIdx)
1489                .MOBILE6Class = CF.Mobile6Class(v)
1490                .HybridType = CF.HybridType(v)
1491            End With ' cafe.vehicles(vidx)
1492            With CAFE.Engines(eIdx)
1493                .EngineOilViscosity = CF.EngineOilViscosity(v)
1494                .ValvesCylinder = CF.ValvesCylinder(v)
1495                .ValveLift = CF.ValveLift(v)
1496                .Aspiration = CF.Aspiration(v)
1497                .ValvetrainDesign = CF.ValveTrainDesign(v)
1498                .Fuel = CF.Fuel(v)
1499                .FuelSystem = CF.FuelSystem(v)
1500                .AirFuelRatio = CF.AirFuelRatio(v)
1501                .MinCompressionRatio = CF.MinCompressionRatio(v)
1502                .MaxCompressionRatio = CF.MaxCompressionRatio(v)
1503            End With ' cafe.engines(eidx)
1504            With CAFE.Transmissions(tIdx)
1505                .Control = CF.Control(v)
1506                .Logic = CF.Logic(v)
1507                .Type = CF.Type(v)
1508                .NumberForwardGears = CF.NumForwardGears(v)
```

```
1509            End With
1510            ' Restore FE, Weight and remove tech cost
1511            CAFE.Vehicles(vIdx).FuelEconomy = CF.VehOrigFE(v)
1512            CAFE.Vehicles(vIdx).CurbWeight = CF.VehOrigWeight(v)
1513            CAFE.Vehicles(vIdx).TechCostByMY(CAFE.GMYI(MY)) -= CF.TechCost(v)
1514            If CF.TechType <> "AERO" And CF.TechType <> "ENGMOD" And CF.TechType <> "TRMOD
    " Then
1515                CAFE.Vehicles(vIdx).TechCostVehTechnologies -= CF.TechCost(v)
1516                CAFE.Vehicles(vIdx).TechImprvVehTechnologies -= CF.TechImprv(v)
1517            End If
1518
1519            ' Check for weight change ..
1520            ' If so, then check for RC change (and STD change)
1521            If CF.VehNewWeight(v) <> CF.VehOrigWeight(v) Then
1522                If CAFE.Scenarios.Scenarios(iSc).LtRegClassAttribute.ToCharArray = "WEIGHT
    " Then
1523                    ' Weight changed AND the RCs are weight based,
1524                    ' so get new RC
1525                    Dim OldRC As Integer = CAFE.Vehicles(vIdx).RegClass
1526                    Dim NewRC As Integer
1527                    NewRC = CAFE.Standards.GetRegClass(iSc, MY, vIdx)
1528                    If NewRC <> OldRC Then
1529                        ' Swap RCs
1530                        CAFE.Mfgs(iMfg).RC_Move(OldRC, NewRC, vIdx)
1531                        Dim OutText As String
1532                        OutText = "Veh Change RC. Old RC: " _
1533                                + OldRC.ToString _
1534                                + " NewRC: " _
1535                                + NewRC.ToString _
1536                                + "  "
1537                        ModellingLogic.ModelErrorsFile.WriteError(OutText)
1538                    End If
1539
1540                    ' Since we've got weight change on a weight based scenario.
1541                    ' we'd better recalc everything (STD, CAFE, Credits and Fines)
1542                    For iRc As Integer = 0 To CAFE.Mfgs(iMfg).nRC - 1
1543                        CAFE.Standards.GetStandard(iSc, MY, iMfg, iRc)
1544                    Next
1545                    CAFE.Mfgs(iMfg).CalcCAFE(MY, True)
1546                    CAFE.Mfgs(iMfg).CalcCredit(iSc, MY, True)
1547                    CAFE.Mfgs(iMfg).CalcFines(iSc, MY, True)
1548                End If
1549            End If
1550
1551            ' restore applies and used states
1552            Dim iMY As Integer = CAFE.GMYI(MY)
1553            For t As Integer = 0 To CAFE.Vehicles(vIdx).TechApplies.Length - 1
1554                CAFE.Vehicles(vIdx).TechApplies(t) = CF.VehTechApplied(v)(t)
1555            Next t
1556            For t As Integer = 0 To CAFE.Vehicles(vIdx).TechUsed.Length - 1
1557                CAFE.Vehicles(vIdx).TechUsed(t) = CF.VehTechUsed(v)(t)
1558                CAFE.Vehicles(vIdx).TechAppliedByModel(t) = CF.VehTechABM(v)(t)
1559                CAFE.Vehicles(vIdx).TechUsedByMY(iMY)(t) = CF.VehTechUsedThisMY(v)(t)
1560                If Array.IndexOf(TechnologiesClass.EngTechnologiesIdx, CF.VehTechUsed(v)
    (t)) <> -1 Then
1561                    CAFE.Engines(CAFE.Vehicles(vIdx).TrueEngineCode).TechUsed(t) = CF.
    VehTechUsed(v)(t)
1562                End If
1563                If Array.IndexOf(TechnologiesClass.TranTechnologiesIdx, CF.VehTechUsed(v)
    (t)) <> -1 Then
1564                    CAFE.Transmissions(CAFE.Vehicles(vIdx).TrueTransmissionCode).TechUsed
    (t) = CF.VehTechUsed(v)(t)
1565                End If
1566                If Array.IndexOf(TechnologiesClass.AeroTechnologiesIdx, CF.VehTechUsed(v)
    (t)) <> -1 Then
1567                    CAFE.Mfgs(CAFE.Vehicles(vIdx).MfgIdx).NamePlates(CAFE.Vehicles(vIdx).
    NamePlateIdx).TechUsed(t) = CF.VehTechUsed(v)(t)
1568                End If
1569            Next t
1570            ' Look for MS1 or MS2
```

```
1571            For t As Integer = 0 To CF.TechIdx.Length - 1
1572                If CF.TechIdx(t) = TechnologiesClass.TechIndexes.tMS_MS1 Then
1573                    ' WE're removing MS1 ... disable MS2
1574                    CAFE.Vehicles(vIdx).TechEnabled(CF.TechIdx(t)) = False
1575                ElseIf CF.TechIdx(t) = TechnologiesClass.TechIndexes.tMS_MS2 Then
1576                    ' We're removing MS2 ... disable MS3
1577                    CAFE.Vehicles(vIdx).TechEnabled(CF.TechIdx(t)) = False
1578                End If
1579            Next
1580        Next v
1581        For t As Integer = 0 To CAFE.Mfgs(iMfg).TechApplied.Length - 1
1582            CAFE.Mfgs(iMfg).TechApplied(t) = CF.MfgTechApplied(t)
1583        Next t
1584        For t As Integer = 0 To CAFE.Mfgs(iMfg).TechExhausted.Length - 1
1585            CAFE.Mfgs(iMfg).TechExhausted(t) = CF.MfgTechExhausted(t)
1586        Next t
1587
1588        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1589        Return True
1590    End Function
1591
1592    Private Sub ReserveCF(ByVal MY As Integer, _
1593                          ByVal iMfg As Integer, _
1594                          ByVal iConv As Integer, _
1595                          ByVal CF As ComplianceFinding, _
1596                          ByVal OF As DebugWriteActions)
1597        '
1598        If IsNothing(Me.CFReserved) Then
1599            ' no compliance finding has been reserved yet--reserve the current one
1600            Me.CFReserved = CF
1601
1602            ' write out log of reserved CF
1603            OF.RecordAction(CF, iConv, , True)
1604        ElseIf CF.TotalTechCost < Me.CFReserved.TotalTechCost Then
1605            ' there is reserved compliance finding, but the new one has a better techCost/ ↙
    fines ratio
1606            ' reserve the new one, overwrting the previous
1607            Me.CFReserved = CF
1608
1609            ' write out log of reserved CF
1610            OF.RecordAction(CF, iConv, , True)
1611        End If
1612
1613        ' set the techs of the new CF as reserved
1614        Me.ReserveTechUsed(MY, iMfg, CF)
1615    End Sub
1616
1617    Private Sub ReserveTechUsed(ByVal MY As Integer, _
1618                                ByVal iMfg As Integer, _
1619                                ByVal CF As ComplianceFinding)
1620        '
1621        Dim i, j As Integer
1622        Dim vIdx, tIdx As Integer
1623
1624        If Not IsNothing(CF) Then
1625            For i = 0 To CF.VehIdx.Length - 1
1626                vIdx = CF.VehIdx(i)
1627
1628                For j = 0 To CF.TechIdx.Length - 1
1629                    tIdx = CF.TechIdx(j)
1630
1631                    If IsTechValid(MY, iMfg, tIdx, vIdx) Then
1632                        ' mark this tech as reserved for this vehicle
1633                        CAFE.Vehicles(vIdx).TechReserved(tIdx) = True
1634                    End If
1635                Next j
1636            Next i
1637        End If
1638    End Sub
1639
```

```
1640        '''
1641        ' Resets CFReserved and any reserved technologies for all vehicles for the specified
1642        ' mfg to prepare for the new compliance finding.
1643        '
1644        Public Sub ResetReservedCF(ByVal iMfg As Integer)
1645            Dim i As Integer
1646            Dim vIdx As Integer
1647
1648            ' reset CFReserved
1649            Me.CFReserved = Nothing
1650
1651            ' scan each veh for this mfg and clear out its ReservedTech array
1652            For i = 0 To CAFE.Mfgs(iMfg).ValidVehicleIndexes.Length - 1
1653                vIdx = CAFE.Mfgs(iMfg).ValidVehicleIndexes(i)
1654                ReDim CAFE.Vehicles(vIdx).TechReserved(TechnologiesClass.nTotalTechnologies -
    1)
1655            Next i
1656        End Sub
1657
1658    #End Region
1659
1660    #Region "Pre/Post Processing"
1661        Private Sub ML_DebugWriteline(ByVal Message As String)
1662            'Debug.WriteLine(Message)
1663        End Sub
1664        ' Scenario pre/post processing. Called before/after the scenario loop
1665        Private Sub PreProcessScenario(ByVal Iteration As Integer)
1666            ML_DebugWriteline("Preprocessing for scenario loop")
1667            ManufacturerClass.PreProcessForScenarioLoop(Iteration)
1668            CAFE.CAFE_Form.InitProgBarScenario(CAFE.Scenarios.nScenarios - 1)
1669            ModellingLogic.NoFinesWarningIssued = False
1670        End Sub
1671        Private Sub PostProcessScenario()
1672            CAFE.CAFE_Form.SetStatusBarScenario(-1)
1673            CAFE.CAFE_Form.SetProgBarScenario(Integer.MaxValue)
1674        End Sub
1675
1676        ' MY pre/post processing. Called before/after the MY loop. Return StartMY
1677        Private Function PreProcessMY(ByVal iScenario As Integer) As Integer
1678            Dim iMY As Integer          ' Index to MYs
1679            Dim StartMY As Integer      ' Starting MY (first valid Model Year)
1680            Dim StartIdxMy As Integer   ' Index to starting MY
1681
1682            ModellingLogic.ModelErrorsFile.StartScenario(iScenario)
1683            CAFE.CAFE_Form.SetProgBarScenario(iScenario)
1684
1685            ManufacturerClass.PreProcessForMYLoop(iScenario)          ' Do MFG level init
    first.
1686
1687            For iMY = 0 To CAFE.ModelYearsSelected.Length - 1
1688                If CAFE.ModelYearsSelected(iMY) Then
1689                    StartMY = CAFE.GetMYearFromIndex(iMY)
1690                    StartIdxMy = iMY
1691                    Exit For
1692                End If
1693            Next
1694            CAFE.CAFE_Form.InitProgBarMY(StartMY, CAFE.GetMYearFromIndex(CAFE.
    ModelYearsSelected.Length - 1))
1695            Return StartMY
1696        End Function
1697        Private Sub PostProcessMY(ByVal iScenario As Integer)
1698            ManufacturerClass.PostProcessForMYLoop(iScenario)
1699            CAFE.CAFE_Form.SetStatusBarMY(-1)
1700            CAFE.CAFE_Form.SetProgBarMY(Integer.MaxValue)
1701            ModellingLogic.ModelErrorsFile.EndScenario()
1702            Application.DoEvents()                                    ' Give event handler a chance
    to breath
1703            Application.DoEvents()
1704        End Sub
1705
```

```
1706    ' Convergence pre/post processing. Call before/after convergence loop
1707    ' Return TRUE if normal run, return FALSE if we're doing a Stage Analysis run.
1708    Private Function PreProcessConvergence(ByVal iScenario As Integer, ByVal MY As
        Integer) As Boolean
1709        ModellingLogic.ModelErrorsFile.StartMY(MY)
1710        ManufacturerClass.PreProcessForConvergenceLoop(iScenario, MY)
1711        If CAFE.OperatingModes.StageAnalysis Then
1712            ' Stage analysis...
1713            If iScenario > 0 Then
1714                ' where there is an entry for FE and CW in the stage analysis data, move
        that to
1715                ' the vehicle's FE and CW
1716                For v As Integer = 0 To CAFE.Vehicles.Length - 1
1717                    If IsNothing(CAFE.Vehicles(v).StageAnalysisFE) = False _
1718                        AndAlso CAFE.Vehicles(v).StageAnalysisFE.Length >= CAFE.GMYI(MY)
        Then
1719                        If CAFE.Vehicles(v).StageAnalysisFE(CAFE.GMYI(MY)) > 0 Then
1720                            CAFE.Vehicles(v).FuelEconomy = CAFE.Vehicles(v).
        StageAnalysisFE(CAFE.GMYI(MY))
1721                        End If
1722                    End If
1723                    If IsNothing(CAFE.Vehicles(v).StageAnalysisCW) = False _
1724                        AndAlso CAFE.Vehicles(v).StageAnalysisCW.Length >= CAFE.GMYI(MY)
        Then
1725                        If CAFE.Vehicles(v).StageAnalysisCW(CAFE.GMYI(MY)) > 0 Then
1726                            CAFE.Vehicles(v).CurbWeight = CAFE.Vehicles(v).StageAnalysisCW
        (CAFE.GMYI(MY))
1727                        End If
1728                    End If
1729                    If IsNothing(CAFE.Vehicles(v).StageAnalysisSales) = False _
1730                        AndAlso CAFE.Vehicles(v).StageAnalysisSales.Length >= CAFE.GMYI
        (MY) Then
1731                        If CAFE.Vehicles(v).StageAnalysisSales(CAFE.GMYI(MY)) > 0 Then
1732                            CAFE.Vehicles(v).Sales(CAFE.GMYI(MY)) = CAFE.Vehicles(v).
        StageAnalysisSales(CAFE.GMYI(MY))
1733                        End If
1734                    End If
1735                Next v
1736            End If
1737            ' output settings -- should be moved to options...
1738            CAFE.OutputSettings.GenerateEffectsReport = True
1739            CAFE.OutputSettings.GenerateEnvironmentalAssessmentReport = False
1740            CAFE.OutputSettings.GenerateIndustryReport = False
1741            CAFE.OutputSettings.GenerateMfgsReport = False
1742            CAFE.OutputSettings.GenerateVehiclesReport = False
1743        End If
1744        Return Not (CAFE.OperatingModes.StageAnalysis And iScenario > 0)
1745    End Function
1746    Private Sub PostProcessConvergence(ByVal iScenario As Integer, ByVal MY As Integer)
1747        ManufacturerClass.PostProcessForConvergenceLoop(iScenario, MY)
1748        CAFE.CAFE_Form.SetStatusBarConv(-1)
1749        ModellingLogic.ModelErrorsFile.EndMY()
1750        Application.DoEvents()                          ' Give event handler a chance
        to breath
1751        Application.DoEvents()
1752    End Sub
1753
1754    ' Mfg pre/post processing. Call before/after mfg loop.
1755    Private Sub PreProcessMfg(ByVal iScenario As Integer, ByVal MY As Integer, ByVal
        iConverge As Integer)
1756        ModellingLogic.ModelErrorsFile.StartConverge(iConverge)
1757        CAFE.CAFE_Form.InitProgBarMfg(CAFE.Mfgs.Length - 1)
1758        ManufacturerClass.PreProcessForMFGLoop(iScenario, MY, iConverge)
1759    End Sub
1760    Private Sub PostProcessMfg(ByVal iScenario As Integer, ByVal MY As Integer, ByVal
        iConverge As Integer)
1761        ManufacturerClass.PostProcessForMFGLoop(iScenario, MY, iConverge)
1762        CAFE.CAFE_Form.InitCAFEProgress(-1, Nothing, Nothing, Nothing)
1763        CAFE.CAFE_Form.SetProgBarMfg(CAFE.Mfgs.Length)
1764        CAFE.CAFE_Form.SetStatusBarMFG("")
```

```
1765        ModellingLogic.ModelErrorsFile.EndConverge()
1766        Application.DoEvents()                                  ' Give event handler a chance ↙
    to breath
1767        Application.DoEvents()
1768    End Sub
1769
1770    ' MFG Init/DeInit routines. Called inside the Mfg loop to initialize certain MFG      ↙
    settings
1771    Private Sub InitMfg(ByVal iScenario As Integer, ByVal MY As Integer, ByVal iConverge  ↙
    As Integer, ByVal iMfg As Integer)
1772        ModellingLogic.ModelErrorsFile.StartMfg(iMfg)
1773
1774        CAFE.CAFE_Form.SetStatusBarMFG(CAFE.Mfgs(iMfg).MfgName)
1775        CAFE.CAFE_Form.SetProgBarMfg(iMfg)
1776        CAFE.Mfgs(iMfg).InitForMFGLoop(iScenario, MY, iConverge)
1777
1778        ReCalcCafeCreditsFines(iScenario, MY, iConverge, iMfg, True)
1779
1780        MyApproach = CAFE.OperatingModes.Approach
1781        MySecondBestOK = False
1782        Dim Willing As Boolean = CAFE.Parameters.GetWillingnessToPayFines(MY, iMfg)
1783
1784        If CAFE.OperatingModes.NoFines Then
1785            NoFines = True
1786        Else
1787            NoFines = Not Willing
1788        End If
1789
1790        ' Initialize the progress bars.
1791        For pRC As Integer = 1 To CAFE.Mfgs(iMfg).RcCAFE.Length - 1
1792            CAFE.CAFE_Form.InitCAFEProgress(pRC, _
1793                                CAFE.Mfgs(iMfg).RcCAFE(pRC), _
1794                                CAFE.Mfgs(iMfg).CalcSTD(iScenario, MY, pRC), _
1795                                CAFE.Mfgs(iMfg).RcFines(pRC))
1796        Next pRC
1797    End Sub
1798    Public Sub DeInitMfg(ByVal iScenario As Integer, ByVal MY As Integer, ByVal iConverge  ↙
    As Integer, ByVal iMfg As Integer)
1799
1800        If DBUG Then
1801            denter(Me, "PostProcessForMFGLoop()")
1802        End If
1803
1804        ReCalcCafeCreditsFines(iScenario, MY, iConverge, iMfg, False)
1805
1806        Application.DoEvents()                                  ' Give event handler a chance ↙
    to breath
1807        Application.DoEvents()
1808    End Sub
1809    Private Sub ReCalcCafeCreditsFines(ByVal iScenario As Integer, _
1810                                       ByVal MY As Integer, _
1811                                       ByVal iConverge As Integer, _
1812                                       ByVal iMfg As Integer, _
1813                                       ByVal RecalcSales As Boolean)
1814        Dim i, j As Integer
1815        Dim nTechs As Integer
1816        Dim vIdx As Integer
1817        Dim MYidx As Integer
1818        Dim Mfg As ManufacturerClass
1819        Dim veh As VehiclesClass
1820
1821        nTechs = TechnologiesClass.nTotalTechnologies
1822        MYidx = CAFE.GMYI(MY)
1823        Mfg = CAFE.Mfgs(iMfg)
1824
1825        ' recalculate everything
1826        Mfg.CalcCAFE(MY)
1827
1828        Dim RcCAFE_Save(Mfg.RcCAFE.Length - 1) As Double
1829        For i = 1 To RcCAFE_Save.Length - 1
```

```
1830                RcCAFE_Save(i) = Mfg.RcCAFE(i)
1831                Mfg.RcCAFE(i) = Math.Round(Mfg.RcCAFE(i), 1)
1832            Next i
1833            Mfg.CalcCredit(iScenario, MY, True, RecalcSales)
1834            Mfg.CalcFines(iScenario, MY)
1835
1836            If Mfg.RcTotalSales(1) > 0 Then
1837                Debug.Write("")
1838            End If
1839            If Mfg.TotalFines <= 0 Then
1840                ' We've reached nirvana!
1841                ' Leave CAFE as-is, and Credits and Fines as they are (based on rounded CAFE)
1842                ' simply return, the fines=0 will be detected and we'll split from mfg loop
1843            Else
1844                ' Not there yet.
1845                'Let's restore CAFE to unrounded, recalc credits and fines and go on our way
1846                For i = 1 To RcCAFE_Save.Length - 1
1847                    ' For domestic autos (RC = 1) or imported autos (RC = 2), if we've reached ↙
        compliance
1848                    ' in that RC, then keep the rounded CAFE.
1849                    If ((i = 1 Or i = 2) And Mfg.RcFines(i) > 0) Or (i > 2) Then
1850                        Mfg.RcCAFE(i) = RcCAFE_Save(i)
1851                    End If
1852                Next i
1853                Mfg.CalcCredit(iScenario, MY, True, False)
1854                Mfg.CalcFines(iScenario, MY)
1855            End If
1856            ' remove NaN's from _Orig and regular objects,
1857            ' and replace them with 0's
1858            Mfg.RemoveNaNs()
1859
1860        End Sub
1861
1862        ' Compliance pre/post processing. Call before/after compliance loop
1863        Private Sub PreProcessCompliance(ByVal iScenario As Integer, ByVal MY As Integer,      ↙
        ByVal iConverge As Integer, ByVal iMfg As Integer)
1864
1865        End Sub
1866        Private Sub PostProcessCompliance(ByVal iScenario As Integer, ByVal MY As Integer,     ↙
        ByVal iConverge As Integer, ByVal iMfg As Integer)
1867
1868            Application.DoEvents()                                  ' Give event handler a chance ↙
        to breath
1869            Application.DoEvents()
1870        End Sub
1871    #End Region
1872
1873    End Class
1874
1875    #Region "ComplianceFinding Class"
1876    Public Class ComplianceFinding
1877        ' This class is used by the compliance loop to communicate findings of technology      ↙
        application
1878        ' to a vehicle or vehicles, as well as to engines and transmissions.
1879
1880        Private Shared CF_Count As Integer = 0
1881        Private DebugPrintFlag As Boolean = False    ' for debug.
1882
1883        ' These are the "outputs" of the ComplianceFinding class. They are generated in       ↙
        CalcEff
1884        Public CFnum As Integer                 ' unique number of this CF
1885        Public Eff As Double                    ' Efficiency Ratio ( < 0.0 considered effective)
1886        Public CafeCostRatio As Double          ' total tech cost / delta cafe ratio
1887        Public FinesCostRatio As Double         ' total tech cost / delta fines
1888        Public NetCost As Double                ' Total cost after factoring fuel savings, etc
1889        Public TotalTechCost As Double          ' Total technology cost
1890        Public TotalTechCostByRc() As Double ' Total technology cost for each reg class
1891        Public NoFinesMode As Boolean           ' Indicates if we're pushing past fines...
1892        Public AltEFF As Double                 ' Using the old style of NetCost/DeltaFines
1893
```

```
1894      ' Over-all indicator of success. If this is false, everything below except Approach is undefined.
1895      Public ChoiceFound As Boolean
1896      Public SecondBest As Boolean        ' If we got an Eff value, but it's not efficient
1897
1898      ' The following are "Inputs" to the ComplianceFinding. They must be set before calling CalcEff
1899      Public Scenario As Integer          ' The scenario number
1900      Public MyIdx As Integer             ' Model year index
1901      Public Convergence As Integer       ' Convergence loop number
1902      Public MfgIdx As Integer            ' Mfg index
1903      Public TechIdx() As Integer         ' Technologies applied
1904      Public TechCost() As Double         ' Technology cost  (indexed by vehicle)
1905      Public TechImprv() As Double        ' FE Improvements (indexed by vehicle)
1906      Public VehIdx() As Integer          ' Vehicles affected
1907      Public VehOrigFE() As Double        ' Original fuel economy on those vehicles
1908      Public VehNewFE() As Double         ' Updated fuel economy on those vehicles
1909      Public VehOrigWeight() As Double    ' CW before technology (index by vehicle)
1910      Public VehDeltaWeight() As Double   ' Delta CurbWeight
1911      Public VehNewWeight() As Double     ' CW after technology (index by vehicle)
1912      Public VehArea() As Double          ' LxW of vehicle (indexed by vehicles)
1913      Public VehSales() As Double         ' Sales of those vehicles
1914      Public EngIdx As Integer            ' Index of engine effected
1915      Public TranIdx As Integer           ' Index of transmission effected
1916      Public NamePlateIdx As Integer      ' Index of nameplate effected
1917      Public CAFE_Before() As Double      ' CAFE for this MFG before applying technology (by RC)
1918      Public CAFE_After() As Double       ' CAFE for this MFG after applying technology (by RC)
1919      Public Fines_Before() As Double     ' Fines for this MFG before applying technology (by RC)
1920      Public Fines_After() As Double      ' Fines for this MFG after applying technology (by RC)
1921
1922      ' Component info.
1923      ' A 'General' component is used when the component is a whole vehicle
1924      ' 'Engine', 'Transmission', and 'Aero' have lists of vehicles associated with them.
1925      ' New 'Engine' and 'Transmission' can be created. If so, the ComponentIdx will represent
1926      ' the IDX of the new, and ComponentOldIdx will have the old.
1927      Public ComponentType As ComponentTypeEnum    ' Type of component (Engine, Transmission, Aero or General)
1928      Public ComponentIdx As Integer               ' If other than General, the [new] component number
1929      Public ComponentCreated As Boolean           ' If a new component created this is true (engines & trans only)
1930      Public ComponentOldIdx As Integer            ' The index of the engine or transmission replaced.
1931
1932      ' Mfg, vehicle, engine and transmission attributes which might get changed if this CF is "installed".
1933      ' These are recorded here in case this CF is "peeled off"
1934      ' Manufacturers:
1935      Public MfgTechApplied(-1) As Double      ' Number of veh with tech applied for phase-in
1936      Public MfgTechExhausted(-1) As Boolean   ' Tech exhausted because of phase-in
1937      ' Vehicles:
1938      Public VehTechApplied(-1)() As Boolean ' TechAPplied for Veh before compliance (index by veh, technologies)
1939      Public VehTechUsed(-1)() As Boolean
1940      Public VehTechABM(-1)() As Boolean
1941      Public VehTechUsedThisMY(-1)() As Boolean
1942      Public Mobile6Class(-1) As String
1943      Public HybridType(-1) As String
1944      ' Engines:
1945      Public EngineOilViscosity(-1) As String
1946      Public ValvesCylinder(-1) As Integer
1947      Public ValveLift(-1) As String
1948      Public Aspiration(-1) As String
1949      Public ValveTrainDesign(-1) As String
1950      Public Fuel(-1) As String
```

```
1951      Public FuelSystem(-1) As String
1952      Public AirFuelRatio(-1) As Double
1953      Public MinCompressionRatio(-1) As Double
1954      Public MaxCompressionRatio(-1) As Double
1955      ' Transmissions:
1956      Public Control(-1) As String
1957      Public Logic(-1) As String
1958      Public Type(-1) As String
1959      Public NumForwardGears(-1) As Integer
1960      Public TechType As String
1961
1962
1963
1964
1965      ' Default constructor
1966      Public Sub New()
1967          If DBUG Then
1968              denter(Me, "New()")
1969          End If
1970
1971          CFnum = ComplianceFinding.CF_Count
1972          ComplianceFinding.CF_Count += 1
1973          ReDim TechIdx(-1)
1974          ChoiceFound = False
1975      End Sub
1976
1977      ' Constructor with basic info (model year, mfg)
1978      Public Sub New(ByVal iSC As Integer, ByVal MY As Integer, ByVal iConverge As Integer, ↙
          _
1979                     ByVal iMfg As Integer, ByVal NoFines As Boolean)
1980          Me.New()
1981
1982          If DBUG Then
1983              denter(Me, "New(iSC, MY, iMfg)")
1984          End If
1985
1986          Scenario = iSC
1987          MyIdx = CAFE.GMYI(MY)
1988          Convergence = iConverge
1989          MfgIdx = iMfg
1990
1991          ' Now fill in some basics
1992          ReDim CAFE_Before(CAFE.Mfgs(iMfg).nRC - 1)
1993          ReDim CAFE_After(CAFE.Mfgs(iMfg).nRC - 1)
1994          ReDim Fines_Before(CAFE.Mfgs(iMfg).nRC - 1)
1995          ReDim Fines_After(CAFE.Mfgs(iMfg).nRC - 1)
1996          For rc As Integer = 0 To CAFE.Mfgs(iMfg).nRC - 1
1997              'CAFE_Before(rc) = CAFE.Mfgs(iMfg).RcCAFE_Orig(rc)
1998              CAFE_Before(rc) = CAFE.Mfgs(iMfg).RcCAFE(rc)
1999              CAFE_After(rc) = Double.NaN
2000              Fines_Before(rc) = CAFE.Mfgs(iMfg).RcFines(rc)
2001              Fines_After(rc) = Double.NaN
2002          Next rc
2003          Me.NoFinesMode = NoFines
2004      End Sub
2005      Public Shared Function IsEfficient(ByRef C As ComplianceFinding)
2006          If IsNothing(C) Then
2007              Return False
2008          Else
2009              Return C.IsEfficient(C.Eff)
2010          End If
2011      End Function
2012
2013      ' Determine (and return) the best efficiency.
2014      ' This takes two compliancefindings and returns the best.
2015      Public Shared Function BestEff(ByRef C1 As ComplianceFinding, ByRef C2 As ↙
          ComplianceFinding) As ComplianceFinding
2016          If DBUG Then
2017              denter(New ComplianceFinding, "BestEff(C1, C2)")
2018          End If
```

```
2019
2020        Dim Carray(1) As ComplianceFinding
2021        Carray(0) = C1
2022        Carray(1) = C2
2023        Return BestEff(Carray)
2024    End Function
2025
2026    ' Determine (and return) the best efficiency.
2027    ' This takes an array of compliancefindings and returns the best.
2028    Public Shared Function BestEff(ByRef C() As ComplianceFinding) As ComplianceFinding
2029        If DBUG Then
2030            denter(New ComplianceFinding, "BestEff(C())")
2031        End If
2032
2033        Dim BestAltEff As Double = Double.MaxValue
2034        Dim ItsEff As Double
2035
2036        Dim beff As ComplianceFinding = Nothing
2037        If IsNothing(C) Then
2038            Return beff
2039        End If
2040        For i As Integer = 0 To C.Length - 1
2041            If IsNothing(C(i)) = False Then
2042                If C(i).ChoiceFound Then
2043                    ' The compliance finding exists and has a valid finding
2044                    ' see if it's the best so far...
2045                    If IsNothing(beff) = False Then
2046                        If C(i).AltEFF < BestAltEff Then
2047                            BestAltEff = C(i).AltEFF
2048                            ItsEff = C(i).Eff
2049                        End If
2050                        If C(i).Eff < beff.Eff Then
2051                            beff = C(i)
2052                        End If
2053                    Else
2054                        ' Beff hasn't been set yet. This is the first valid find
2055                        beff = C(i)
2056                    End If
2057                End If
2058            End If
2059        Next
2060        'If (IsNothing(beff) = False AndAlso beff.AltEFF > BestAltEff) Then
2061        '    Debug.WriteLine("eff/alteff mismatch: eff: " + beff.Eff.ToString +
2062        '    " Aeff: " + beff.AltEFF.ToString + " bAeff: " + BestAltEff.ToString + "
    ItsEff: " + ItsEff.ToString)
2063        'End If
2064        Return beff
2065    End Function
2066
2067    ' Determine (and return) the second best efficiency (lowest eff, even of > 0).
2068    ' This takes two compliancefindings and returns the best.
2069    Public Shared Function SecondBestEff(ByRef C1 As ComplianceFinding, ByRef C2 As
    ComplianceFinding) As ComplianceFinding
2070        If DBUG Then
2071            denter(New ComplianceFinding, "SecondBestEff(C1, C2)")
2072        End If
2073
2074        Dim Carray(1) As ComplianceFinding
2075        Carray(0) = C1
2076        Carray(1) = C2
2077        Return SecondBestEff(Carray)
2078    End Function
2079
2080    ' Determine (and return) the second best efficiency (lowest eff, even of > 0).
2081    ' This takes an array of compliancefindings and returns the best.
2082    Public Shared Function SecondBestEff(ByRef C() As ComplianceFinding) As
    ComplianceFinding
2083        If DBUG Then
2084            denter(New ComplianceFinding, "SecondBestEff(C())")
2085        End If
```

```
2086
2087        Dim beff As ComplianceFinding = Nothing
2088        If IsNothing(C) Then
2089            Return beff
2090        End If
2091        For i As Integer = 0 To C.Length - 1
2092            If IsNothing(C(i)) = False Then
2093                If C(i).ChoiceFound Or C(i).SecondBest Then
2094                    ' The compliance finding exists and has a valid finding
2095                    ' see if it's the best so far...
2096                    If IsNothing(beff) = False Then
2097                        If C(i).Eff < beff.Eff Then
2098                            beff = C(i)
2099                        End If
2100                    Else
2101                        ' Beff hasn't been set yet. This is the first valid find
2102                        beff = C(i)
2103                    End If
2104                End If
2105            End If
2106        Next
2107        Return beff
2108    End Function
2109    ' Caclculate EFF based on input.
2110    Public Function CalcEff() As Double
2111        If DBUG Then
2112            denter(Me, "CalcEff()")
2113        End If
2114
2115        Dim printflag As Boolean = False
2116        Dim printstring As String
2117
2118        Dim AltNC As Double             ' for alteff calculations...
2119
2120        Dim Eff As Double = 0
2121        Dim EffDenominator As Double = 0        ' accumulator for EFF denominator (KG:  03 ↙
    /04/05)
2122        Dim EffNumerator As Double = 0          ' accumulator for Eff numerator (KG:  03/ ↙
    04/05)
2123
2124        Dim RC_dCafe(CAFE.Mfgs(MfgIdx).nRC - 1)         ' Delta Cafe by RC
2125        Dim RC_dFines(CAFE.Mfgs(MfgIdx).nRC - 1)        ' Delta Fines by RC (calculated    ↙
    differently than RC_dF)
2126        Dim RC_CafeRatio(CAFE.Mfgs(MfgIdx).nRC - 1)     ' (TechCost / Cafe) ratio by rc
2127        Dim RC_FinesRatio(CAFE.Mfgs(MfgIdx).nRC - 1)    ' (TechCost / Fines) ratio by rc
2128
2129        Dim RC_dF(11) As Double         ' Delta Fine by RC
2130        Dim RC_dRPE(11) As Double       ' Delta RPE by RC
2131        Dim RC_FC_Savings(11) As Double ' Fuel Consumption Savings by RC
2132        Dim RC_Affected_Sales(11) As Double ' Number of units affected by current action   ↙
    (KG:  03/04/05)
2133
2134        Dim veh() As Integer
2135        Dim c() As Double   ' costs
2136        Dim bfe() As Double
2137        Dim nfe() As Double
2138        Dim nv As Integer               ' number of vehicles (per RC)
2139        Dim TotalRegulatedVehicles As Integer = 0       ' Total regulated vehicles in this ↙
     CF
2140
2141        ' Gather the values by RC.
2142        ReDim Me.TotalTechCostByRc(CAFE.Mfgs(MfgIdx).nRC - 1)
2143        TotalTechCost = 0
2144        NetCost = 0
2145
2146        ' First check to see if there are some vehicles in regulated classes
2147        Dim Reg0 As Boolean = False
2148        Dim RegN0 As Boolean = False
2149        For v As Integer = 0 To VehIdx.Length - 1
2150            If CAFE.Vehicles(VehIdx(v)).RegClass = 0 Then
```

```
2151                RegO = True
2152            Else
2153                RegNO = True
2154            End If
2155        Next v
2156
2157        If RegNO = False Then
2158            ' No vehicles in regulated classes.  Shut off this ComplianceFinding
2159            Me.ChoiceFound = False
2160            For v As Integer = 0 To Me.VehIdx.Length - 1
2161                Dim vidx = VehIdx(v)
2162                For t As Integer = 0 To Me.TechIdx.Length - 1
2163                    Dim tidx = TechIdx(t)
2164                    CAFE.Vehicles(vidx).TechIgnored(tidx) = True
2165                Next t
2166            Next v
2167            Return Double.PositiveInfinity
2168        End If
2169
2170        ' Loop through RCs
2171        For rc As Integer = 0 To CAFE.Mfgs(MfgIdx).nRC - 1
2172            nv = 0
2173            ' Capture vehicles in this RC
2174            For v As Integer = 0 To VehIdx.Length - 1
2175                If CAFE.Vehicles(VehIdx(v)).RegClass = rc Then
2176                    ReDim Preserve veh(nv)
2177                    ReDim Preserve c(nv)
2178                    ReDim Preserve bfe(nv)
2179                    ReDim Preserve nfe(nv)
2180                    veh(nv) = VehIdx(v)
2181                    c(nv) = TechCost(v)
2182                    bfe(nv) = VehOrigFE(v)
2183                    nfe(nv) = VehNewFE(v)
2184                    nv += 1
2185                    TotalRegulatedVehicles += 1
2186                End If
2187            Next v
2188            ' now calculate EFF for this RC, adding it to the accumulated EFF
2189            For v As Integer = 0 To nv - 1
2190                Dim Sales As Double = CAFE.Vehicles(veh(v)).Sales(MyIdx)
2191                ' get fuel type
2192                Dim FuelType As String = LCase(CAFE.Engines(CAFE.Vehicles(veh(v)).
    TrueEngineCode).Fuel)
2193                Dim KmyVal As Double
2194                If FuelType = "d" Then
2195                    ' diesel
2196                    KmyVal = CAFE.Parameters.Kmy(MyIdx).Diesel
2197                Else
2198                    ' gas
2199                    KmyVal = CAFE.Parameters.Kmy(MyIdx).Gasoline
2200                End If
2201                RC_Affected_Sales(rc) += Sales ' KG:  03/04/05
2202                RC_dRPE(rc) += (Sales * c(v))
2203                RC_FC_Savings(rc) += (Sales * KmyVal * ((1 / bfe(v)) - (1 / nfe(v))))
2204                TotalTechCost += (Sales * c(v))
2205                TotalTechCostByRc(rc) += (Sales * c(v))
2206                NetCost += (c(v) - RC_FC_Savings(rc))
2207                AltNC += Sales * (c(v) - KmyVal * ((1 / bfe(v)) - (1 / nfe(v))))
2208                If printflag Then
2209                    printstring = "Veh: " + veh(v).ToString + " BFE: " + bfe(v).ToString +
     " NFE: " + nfe(v).ToString + " S: " + Sales.ToString _
2210                    + " C: " + c(v).ToString + " D_RPE: " + RC_dRPE(rc).ToString + " FCS:
    " + RC_FC_Savings(rc).ToString
2211                End If
2212            Next v
2213            If rc > 0 Then
2214                RC_dF(rc) = CAFE.Mfgs(MfgIdx).RcFines(rc) - Fines_After(rc)
2215            Else
2216                RC_dF(rc) = 0
2217            End If
```

```
2218
2219            ' calculate delta cafe and ratio for this rc
2220            RC_dCafe(rc) = Me.CAFE_After(rc) - Me.CAFE_Before(rc)
2221            RC_dFines(rc) = Me.Fines_After(rc) - Me.Fines_Before(rc)
2222            RC_CafeRatio(rc) = Me.TotalTechCostByRc(rc) / RC_dCafe(rc)
2223
2224            ' calculate overall "tech cost to delta cafe" and "tech cost to delta fines" ratios for this CF
2225            Me.CafeCostRatio = 0
2226            Me.FinesCostRatio = 0
2227            If Not Double.IsNaN(RC_CafeRatio(rc)) Then
2228                Me.CafeCostRatio += RC_CafeRatio(rc) * (CAFE.Mfgs(MfgIdx).RcTotalSales(rc) / CAFE.Mfgs(MfgIdx).TotalSales)
2229                Me.FinesCostRatio += RC_FinesRatio(rc) * (CAFE.Mfgs(MfgIdx).RcTotalSales(rc) / CAFE.Mfgs(MfgIdx).TotalSales)
2230            End If
2231        Next rc
2232
2233        Dim DeltaFine As Double = 0
2234        ' now loop and tally total DeltaFine
2235        For rc As Integer = 1 To CAFE.Mfgs(MfgIdx).nRC - 1
2236            DeltaFine += RC_dF(rc)
2237        Next
2238
2239        ' If we've changed the fine, then we've acomplished something...
2240        ' calc effectiveness
2241        If DeltaFine <> 0 Then
2242            For rc As Integer = 0 To CAFE.Mfgs(MfgIdx).nRC - 1
2243                If DeltaFine < 0 And ModellingLogic.NoFinesWarningIssued = False Then
2244                    If Array.IndexOf(Me.TechIdx, TechnologiesClass.TechIndexes.tMS_MSX) < 0 Then
2245                        'MsgBox("DeltaFine is < 0!!!")
2246                        ModellingLogic.NoFinesWarningIssued = True
2247                        ModellingLogic.ModelErrorsFile.ErrorWithComplianceFinding(Me, "Negative Delta Fine")
2248                    End If
2249                End If
2250                If DeltaFine >= 0 Or DeltaFine < 0 And Array.IndexOf(TechIdx, TechnologiesClass.TechIndexes.tMS_MSX) < 0 Then
2251                    EffDenominator += RC_Affected_Sales(rc) ' KG:  03/04/05
2252                    EffNumerator += (RC_dRPE(rc) - RC_dF(rc) - RC_FC_Savings(rc)) ' KG:  03/04/05
2253                    AltEFF += (AltNC / DeltaFine)
2254                    If printflag Then
2255                        printstring += " D_Fine: " + DeltaFine.ToString + " Eff: " + Eff.ToString
2256                        Debug.WriteLine(printstring)
2257                    End If
2258                Else
2259                    EffNumerator += 0    ' this line here just for illustration. (KG:  03/04/05)
2260                End If
2261            Next rc
2262
2263            ' set the efficient value for this compliance
2264            If TotalRegulatedVehicles > 0 AndAlso EffDenominator > 0 Then ' KG:  03/04/05
2265                Eff = EffNumerator / EffDenominator ' KG:  03/04/05
2266                Me.Eff = Eff
2267                Me.ChoiceFound = True
2268            Else
2269                Me.Eff = Double.PositiveInfinity
2270                Me.ChoiceFound = False
2271            End If
2272            ' then return it
2273        Else
2274            ' Delta fine is 0...
2275            ' We're going to discard this CF and set TechIgnore for the vehicles/techs
2276            Me.ChoiceFound = False
2277            Me.Eff = Double.PositiveInfinity
2278            For v As Integer = 0 To Me.VehIdx.Length - 1
```

```
2279                Dim vidx = VehIdx(v)
2280                For t As Integer = 0 To Me.TechIdx.Length - 1
2281                    Dim tidx = TechIdx(t)
2282                    CAFE.Vehicles(vidx).TechIgnored(tidx) = True
2283                Next t
2284            Next v
2285        End If
2286
2287        Return Eff
2288    End Function
2289
2290    ' Set the ChoiceFound member based on me.IsEfficient
2291    Public Sub SetChoiceFound()
2292        If DBUG Then
2293            denter(Me, "SetChoiceFound()")
2294        End If
2295
2296        If Double.IsNaN(Eff) = False And Double.IsInfinity(Eff) = False Then
2297            ChoiceFound = IsEfficient()
2298            SecondBest = Not IsEfficient()
2299        Else
2300            ChoiceFound = False
2301            SecondBest = False
2302        End If
2303    End Sub
2304
2305    ' Return T/F to indicate if this eff value is efficient
2306    Public Shared Function IsEfficient(ByVal EffVal As Double) As Boolean
2307        If DBUG Then
2308            denter(New ComplianceFinding, "IsEfficient(EffVal)")
2309        End If
2310
2311        If Double.IsNaN(EffVal) Then
2312            Return False
2313        End If
2314        Return (EffVal < 0)
2315    End Function
2316
2317    Public Function IsEfficient() As Boolean
2318        If DBUG Then
2319            denter(Me, "IsEfficient()")
2320        End If
2321
2322        Return ComplianceFinding.IsEfficient(Me.Eff)
2323    End Function
2324
2325    ' Add a vehicle to the class
2326    ' - vIdx:     The vehicle we're adding with the following information:
2327    ' - tidx:     technologies being applied to the vehicle
2328    ' - tCost:    cost of the technologies
2329    ' - vOrigFE:  fuel economy before the technology
2330    ' - vNewFE:   fuel economy after the technology
2331    ' - vOrigCW:  Original curb weight ov vehicle
2332    ' - vDeltaCW: Change in curb weight (we calc NewCW)
2333    ' - vArea:    Area (footprint, L*W, shadow, etc) of vehicle
2334    ' - vSales:   sales of the vehicle
2335    Public Sub AddVehicleInfo(ByVal vIdx As Integer, _
2336                              ByVal tIdx() As Integer, _
2337                              ByVal tCost As Double, _
2338                              ByVal tImprv As Double, _
2339                              ByVal vOrigFE As Double, _
2340                              ByVal vNewFE As Double, _
2341                              ByVal vOrigCW As Double, _
2342                              ByVal vDeltaCW As Double, _
2343                              ByVal vArea As Double, _
2344                              ByVal vSales As Double)
2345        If DBUG Then
2346            denter(Me, "AddVehicleInfo(vIdx, tIdx, tCost, vOrigFE, vNewFE, vOrigW, vNewW,
    vArea, vSales)")
2347        End If
```

```
2348
2349        ' Determine size of vehidx array, and redim make it grow.
2350        Dim nSize As Integer = 0
2351        If Not IsNothing(VehIdx) Then    ' need to test if VehIdx previously existed
2352            nSize = VehIdx.Length        ' if it did, use its length for nSize
2353        End If
2354        ReDim Preserve VehIdx(nSize)
2355        ReDim Preserve TechCost(nSize)
2356        ReDim Preserve TechImprv(nSize)
2357        ReDim Preserve VehOrigFE(nSize)
2358        ReDim Preserve VehNewFE(nSize)
2359        ReDim Preserve VehOrigWeight(nSize)
2360        ReDim Preserve VehDeltaWeight(nSize)
2361        ReDim Preserve VehNewWeight(nSize)
2362        ReDim Preserve VehArea(nSize)
2363        ReDim Preserve VehSales(nSize)
2364
2365        ' Now store the new data
2366        VehIdx(nSize) = vIdx
2367        TechCost(nSize) = tCost
2368        TechImprv(nSize) = tImprv
2369        VehOrigFE(nSize) = vOrigFE
2370        VehNewFE(nSize) = vNewFE
2371        VehOrigWeight(nSize) = vOrigCW
2372        VehDeltaWeight(nSize) = vDeltaCW
2373        VehNewWeight(nSize) = vOrigCW - vDeltaCW
2374        VehArea(nSize) = vArea
2375        VehSales(nSize) = vSales
2376
2377
2378        ' Deal with technology indexes. Identify "new" technologies to add
2379        'If tIdx.Length > 1 Then
2380        '    Debug.WriteLine("TechTable: " + tIdx.Length.ToString + "  already have: " +
    Me.TechIdx.Length.ToString)
2381        'End If
2382        Array.Sort(tIdx)
2383        Dim t As Integer
2384        Dim tPos() As Boolean
2385        ReDim tPos(tIdx.Length - 1)
2386        For t = 0 To tIdx.Length - 1
2387            If Array.IndexOf(Me.TechIdx, tIdx(t)) = -1 Then
2388                tPos(t) = True
2389            End If
2390        Next t
2391
2392        ' Now add any technologies which are new to the list
2393        For t = 0 To tPos.Length - 1
2394            If tPos(t) Then
2395                Dim l As Integer
2396                l = Me.TechIdx.Length
2397                ReDim Preserve Me.TechIdx(l)
2398                Me.TechIdx(l) = tIdx(t)
2399            End If
2400        Next t
2401        ' resort the technologies
2402        Array.Sort(Me.TechIdx)
2403
2404        ' Copy vehicle (and engine and transmission) attributes
2405
2406        ' Copy vehicle (incl engine and transmission) attributes -- as well as TechApplies
     settings.
2407        ' This ~might~ be used later if we're peeling off technologies in subsequent
    convergence loops.
2408        ReDim Me.MfgTechApplied(CAFE.Mfgs(Me.MfgIdx).TechApplied.Length - 1)
2409        For t = 0 To Me.MfgTechApplied.Length - 1
2410            Me.MfgTechApplied(t) = CAFE.Mfgs(Me.MfgIdx).TechApplied(t)
2411        Next t
2412        For t = 0 To Me.MfgTechExhausted.Length - 1
2413            Me.MfgTechExhausted(t) = CAFE.Mfgs(Me.MfgIdx).TechExhausted(t)
2414        Next t
```

```
2415            ReDim Me.MfgTechExhausted(CAFE.Mfgs(Me.MfgIdx).TechExhausted.Length - 1)
2416            ReDim Preserve Me.VehTechApplied(nSize)
2417            ReDim Preserve Me.VehTechUsed(nSize)
2418            ReDim Preserve Me.VehTechABM(nSize)
2419            ReDim Preserve Me.VehTechUsedThisMY(nSize)
2420            ReDim Preserve Me.VehTechApplied(nSize)(CAFE.Vehicles(vIdx).TechApplies.Length -
        1)
2421            ReDim Preserve Me.VehTechUsed(nSize)(CAFE.Vehicles(vIdx).TechUsed.Length - 1)
2422            ReDim Preserve Me.VehTechABM(nSize)(CAFE.Vehicles(vIdx).TechAppliedByModel.Length
        - 1)
2423            ReDim Preserve Me.VehTechUsedThisMY(nSize)(CAFE.Vehicles(vIdx).TechUsed.Length -
        1)
2424            For t = 0 To CAFE.Vehicles(vIdx).TechApplies.Length - 1
2425                Me.VehTechApplied(nSize)(t) = CAFE.Vehicles(vIdx).TechApplies(t)
2426            Next t
2427            For t = 0 To CAFE.Vehicles(vIdx).TechUsed.Length - 1
2428                Me.VehTechUsed(nSize)(t) = CAFE.Vehicles(vIdx).TechUsed(t)
2429                Me.VehTechABM(nSize)(t) = CAFE.Vehicles(vIdx).TechAppliedByModel(t)
2430                Me.VehTechUsedThisMY(nSize)(t) = CAFE.Vehicles(vIdx).TechUsedByMY(MyIdx)(t)
2431            Next t
2432            ReDim Preserve Me.Mobile6Class(nSize)
2433            ReDim Preserve Me.HybridType(nSize)
2434            ReDim Preserve Me.EngineOilViscosity(nSize)
2435            ReDim Preserve Me.ValvesCylinder(nSize)
2436            ReDim Preserve Me.ValveLift(nSize)
2437            ReDim Preserve Me.Aspiration(nSize)
2438            ReDim Preserve Me.ValveTrainDesign(nSize)
2439            ReDim Preserve Me.Fuel(nSize)
2440            ReDim Preserve Me.FuelSystem(nSize)
2441            ReDim Preserve Me.AirFuelRatio(nSize)
2442            ReDim Preserve Me.MinCompressionRatio(nSize)
2443            ReDim Preserve Me.MaxCompressionRatio(nSize)
2444            ReDim Preserve Me.Control(nSize)
2445            ReDim Preserve Me.Logic(nSize)
2446            ReDim Preserve Me.Type(nSize)
2447            ReDim Preserve Me.NumForwardGears(nSize)
2448
2449            Dim ex As Integer = CAFE.Vehicles(vIdx).TrueEngineCode
2450            Dim tx As Integer = CAFE.Vehicles(vIdx).TrueTransmissionCode
2451
2452            Me.Mobile6Class(nSize) = CAFE.Vehicles(vIdx).MOBILE6Class
2453            Me.HybridType(nSize) = CAFE.Vehicles(vIdx).HybridType
2454            Me.EngineOilViscosity(nSize) = CAFE.Engines(ex).EngineOilViscosity
2455            Me.ValvesCylinder(nSize) = CAFE.Engines(ex).ValvesCylinder
2456            Me.ValveLift(nSize) = CAFE.Engines(ex).ValveLift
2457            Me.Aspiration(nSize) = CAFE.Engines(ex).Aspiration
2458            Me.ValveTrainDesign(nSize) = CAFE.Engines(ex).ValvetrainDesign
2459            Me.FuelSystem(nSize) = CAFE.Engines(ex).Fuel
2460            Me.FuelSystem(nSize) = CAFE.Engines(ex).FuelSystem
2461            Me.AirFuelRatio(nSize) = CAFE.Engines(ex).AirFuelRatio
2462            Me.MinCompressionRatio(nSize) = CAFE.Engines(ex).MinCompressionRatio
2463            Me.MaxCompressionRatio(nSize) = CAFE.Engines(ex).MaxCompressionRatio
2464            Me.Control(nSize) = CAFE.Transmissions(tx).Control
2465            Me.Logic(nSize) = CAFE.Transmissions(tx).Logic
2466            Me.Type(nSize) = CAFE.Transmissions(tx).Type
2467            Me.NumForwardGears(nSize) = CAFE.Transmissions(tx).NumberForwardGears
2468
2469        End Sub
2470
2471        ' Add RC oriented information to the class
2472        Public Sub AddRCInfo(ByVal CAFE__After() As Double, ByVal Fines__After() As Double)
2473            If DBUG Then
2474                denter(Me, "AddVehicleInfo(CAFE__After(), Fines__After())")
2475            End If
2476
2477            For rc As Integer = 0 To CAFE__After.Length - 1
2478                CAFE_After(rc) = CAFE__After(rc)
2479                Fines_After(rc) = Fines__After(rc)
2480            Next rc
2481        End Sub
```

```
2482 End Class
2483 #End Region
2484
2485 #Region "MonteCarloClass"
2486 Public Class MonteCarloClass
2487     Inherits ModellingLogic
2488     Private _Iterations As Integer
2489
2490     Public Sub New(ByVal Iterations As Integer)
2491         MyBase.New()
2492         Dim IterationTime As DateTime
2493         _Iterations = Iterations
2494         IterationTime = CAFE.CAFE_Form.ModelStartTime
2495         Dim i As Integer
2496
2497         'Dim MyDebugFile As New StreamWriter("C:\VanSchalkwyk\Foo.out")
2498         'Dim MyWriter As New TextWriterTraceListener(MyDebugFile)
2499         'Debug.Listeners.Clear()
2500         'Debug.Listeners.Add(MyWriter)
2501
2502         For i = 0 To _Iterations - 1
2503             CAFE.CurrentIteration = i
2504             Debug.WriteLine("Starting iteration: " + i.ToString)
2505             Debug.Indent()
2506             CAFE.SummaryOutput = Nothing
2507             CAFE.SummaryOutput = New SummaryInfoClass(i)
2508             MyBase.Begin(i)
2509             CAFE.SummaryOutput.Close()
2510             Dim ThisTime As DateTime = DateTime.Now
2511             Dim TimeDiff As TimeSpan = ThisTime.Subtract(IterationTime)
2512             Dim displaytime As TimeSpan = New TimeSpan(TimeDiff.Days, _
2513                                                        TimeDiff.Hours, _
2514                                                        TimeDiff.Minutes, _
2515                                                        TimeDiff.Seconds)
2516             Debug.Unindent()
2517             Debug.WriteLine("Iteration: " + i.ToString + "  duration: " + displaytime.
    ToString)
2518             'ModellingLogic.ModelErrorsFile.WriteString("Iteration:," + i.ToString + ",
    duration:," + displaytime.ToString)
2519             IterationTime = DateTime.Now
2520         Next i
2521         Debug.WriteLine("Done!")
2522     End Sub
2523
2524 End Class
2525 #End Region
2526
2527 #Region "CF_StackClass"
2528 Public Class CF_StackClass
2529     Public CFstack(-1) As ComplianceFinding
2530     Public Count As Integer
2531     Public ByRC(11) As ArrayList
2532     Public PureRC(11) As ArrayList
2533     'Public PureCars As ArrayList
2534     'Public PureTrucks As ArrayList
2535
2536     Public Sub New()
2537         Count = 0
2538         For i As Integer = 0 To 11
2539             ByRC(i) = New ArrayList
2540             ByRC(i).Clear()
2541             PureRC(i) = New ArrayList
2542             PureRC(i).Clear()
2543         Next
2544         'PureCars = New ArrayList
2545         'PureCars.Clear()
2546         'PureTrucks = New ArrayList
2547         'PureTrucks.Clear()
2548     End Sub
2549     Public Sub push(ByVal CF As ComplianceFinding)
```

```
2550        Dim idx As Integer = CFstack.Length
2551        ReDim Preserve CFstack(idx)
2552        Dim WhichRC(11) As Boolean
2553        Dim ispurerc As Boolean = True
2554        Dim ispurecar As Boolean = True
2555        Dim ispuretruck As Boolean = True
2556        Dim rc As Integer
2557        Dim lastrc As Integer
2558        CFstack(idx) = CF
2559        Try
2560            For v As Integer = 0 To CF.VehIdx.Length - 1
2561
2562            Next
2563        Catch ex As Exception
2564            Debug.WriteLine(ex.Message)
2565        End Try
2566        For v As Integer = 0 To CF.VehIdx.Length - 1
2567            Dim vehidx As Integer = CF.VehIdx(v)
2568            rc = CAFE.Vehicles(vehidx).RegClass
2569            WhichRC(rc) = True
2570            If v > 0 And rc <> lastrc Then
2571                ispurerc = False
2572            End If
2573            lastrc = rc
2574            If rc = 1 Or rc = 2 Then
2575                ispuretruck = False
2576            Else
2577                ispurecar = False
2578            End If
2579        Next v
2580        For r As Integer = 0 To 11
2581            If WhichRC(r) Then
2582                ByRC(r).Add(idx)
2583            End If
2584        Next r
2585        If ispurerc Then
2586            PureRC(rc).Add(idx)
2587        End If
2588        'If ispurecar Then
2589        '    PureCars.Add(idx)
2590        'End If
2591        'If ispuretruck Then
2592        '    PureTrucks.Add(idx)
2593        'End If
2594        Count += 1
2595    End Sub
2596    Public Function Pop() As ComplianceFinding
2597        Dim rCF As ComplianceFinding = Nothing
2598        Dim i As Integer
2599        Dim idx As Integer
2600        If Count > 0 Then
2601            For i = CFstack.Length - 1 To 0 Step -1
2602                If IsNothing(rCF) And Not IsNothing(CFstack(i)) Then
2603                    idx = i
2604                    rCF = CFstack(i)
2605                    CFstack(i) = Nothing
2606                End If
2607            Next i
2608
2609            ' remove this element from associated lists
2610            For r As Integer = 0 To 11
2611                ByRC(r).Remove(idx)
2612                PureRC(r).Remove(idx)
2613            Next r
2614            'PureCars.Remove(idx)
2615            'PureTrucks.Remove(idx)
2616            Count = Count - 1
2617            ReDim Preserve CFstack(Count)
2618        End If
2619        Return rCF
```

```
2620        End Function
2621        Public Function Peek() As ComplianceFinding
2622            Dim rCF As ComplianceFinding = Nothing
2623            Dim i As Integer
2624            Dim idx As Integer
2625            If Count > 0 Then
2626                For i = Me.CFstack.Length - 1 To 0 Step -1
2627                    If IsNothing(rCF) Then
2628                        idx = i
2629                        rCF = CFstack(i)
2630                        CFstack(i) = Nothing
2631                    End If
2632                Next i
2633            End If
2634            Return rCF
2635        End Function
2636        Public Function PopByRC(ByVal rc As Integer) As ComplianceFinding
2637            Dim rCF As ComplianceFinding = Nothing
2638            Dim index As Integer
2639            ' first check the pure list
2640            If PureRC(rc).Count > 0 Then
2641                If PureRC(rc).Count > 0 Then
2642                    index = PureRC(rc)(PureRC(rc).Count - 1)
2643                    rCF = Yank(index)
2644                End If
2645            ElseIf rc = 1 Or rc = 2 Then
2646                'If PureCars.Count > 0 Then
2647                '    index = PureCars(PureCars.Count - 1)
2648                '    rCF = Yank(index)
2649                'End If
2650                'ElseIf Not (rc = 1 Or rc = 2) Then
2651                'If PureTrucks.Count > 0 Then
2652                '        index = PureTrucks(PureTrucks.Count - 1)
2653                '    rCF = Yank(index)
2654                'End If
2655            ElseIf ByRC(rc).Count > 0 Then
2656                If ByRC(rc).Count > 0 Then
2657                    index = ByRC(rc)(ByRC(rc).Count - 1)
2658                    rCF = Yank(index)
2659                End If
2660            End If
2661
2662            Return rCF
2663        End Function
2664        Public Function PeekByRC(ByVal rc As Integer) As ComplianceFinding
2665            Dim rCF As ComplianceFinding = Nothing
2666            Dim index As Integer
2667            ' first check the pure list
2668            If PureRC(rc).Count > 0 Then
2669                index = PureRC(rc)(PureRC(rc).Count - 1)
2670                If index < Count Then
2671                    rCF = Me.CFstack(index)
2672                End If
2673                'ElseIf rc = 1 Or rc = 2 Then
2674                '    index = PureCars(PureCars.Count - 1)
2675                '    If index < Count Then
2676                '        rCF = Me.CFstack(index)
2677                '    End If
2678                'ElseIf Not (rc = 1 Or rc = 2) Then
2679                '    index = Puretrucks(Puretrucks.Count - 1)
2680                '    If index < Count Then
2681                '        rCF = Me.CFstack(index)
2682                '    End If
2683            ElseIf ByRC(rc).Count > 0 Then
2684                index = ByRC(rc)(ByRC(rc).Count - 1)
2685                If index < Count Then
2686                    rCF = Me.CFstack(index)
2687                End If
2688            End If
2689            Return rCF
```

```
2690      End Function
2691      Public Function Yank(ByVal index As Integer) As ComplianceFinding
2692          Dim rCF As ComplianceFinding = Nothing
2693          If index >= 0 And index < CFstack.Length Then
2694              rCF = CFstack(index)
2695              If IsNothing(rCF) Then
2696                  Debug.WriteLine("Yanking a null entry")
2697              End If
2698              CFstack(index) = Nothing
2699              For i As Integer = index To CFstack.Length - 2
2700                  CFstack(i) = CFstack(i + 1)
2701              Next
2702              ReDim Preserve CFstack(CFstack.Length - 2)
2703              Me.Count -= 1
2704              ' remove this element from associated lists
2705              For r As Integer = 0 To 11
2706                  ByRC(r).Remove(index)
2707                  PureRC(r).Remove(index)
2708              Next r
2709              'PureCars.Remove(index)
2710              'PureTrucks.Remove(index)
2711          End If
2712          Return rCF
2713      End Function
2714      Public Sub Clear()
2715          Count = 0
2716          For i As Integer = 0 To 11
2717              ByRC(i).Clear()
2718              PureRC(i).Clear()
2719          Next
2720          'PureCars.Clear()
2721          'PureTrucks.Clear()
2722      End Sub
2723  End Class
2724  #End Region
2725
2726  #Region "TestInfo Class"
2727  Public Class TestInfo
2728      ' This class is used when testing a technology with a vehicle or vehicles.
2729      ' It will contain the information for routines such as CalcFINE, GetStandard, etc
2730      '
2731      ' It's also used to pass back information from 'TouchTechnologies' to give a profile ↙
         of
2732      ' vehicle/technology combinations
2733      '
2734      Public MfgIdx As Integer                ' Index of the manufacturer
2735      Public nTestVehicles As Integer         ' Number of vehicles involved
2736      Public TestVehIdx() As Integer          ' Indexes of the vehicles
2737      Public TestVehFE() As Double            ' Fuel Economy of those vehicles
2738      Public TestArea() As Double             ' Area (length x width) of those vehicles
2739      Public TestWeight() As Double           ' Weight (CW) of those vehicles
2740  End Class
2741  #End Region
2742
2743  #Region "TechProfile Class"
2744  Public Class TechProfile
2745      Public TechIdx() As Integer             ' The technologies
2746      Public VehIdx() As Integer              ' The vehicles
2747      Public RC As Integer                    ' The regulatory class of the vehicles
2748      Public NeedsNewComponent As Boolean ' Must add an engine or a transmission for this ↙
         profile
2749      Public NewComponentType As ComponentTypeEnum
2750
2751      ' Descriptive properties of the profile.
2752      Public Orig As Boolean                  ' "Original". This profile is used as the base in ↙
         comparison
2753      Public DiffRC As Boolean                ' the RC is different from the "Original"
2754      Public DiffEnab As Boolean              ' the technology enable states differ from ↙
         "Original"
2755      Private _IsAuto As Boolean              ' the vehicles in this profile are automobiles
```

```
2756    Private _IsLightTruck As Boolean     ' the vehicles in this profile are light trucks
2757    Private _IsUnregulated As Boolean    ' the vehicles in this profile are unregulated
2758
2759    Public Sub New()
2760        VehIdx = Nothing
2761        TechIdx = Nothing
2762        RC = -1
2763    End Sub
2764    Public Sub New(ByVal srcProfile As TechProfile)
2765        ReDim Me.TechIdx(srcProfile.TechIdx.Length - 1)
2766        ReDim Me.VehIdx(srcProfile.VehIdx.Length - 1)
2767        For t As Integer = 0 To srcProfile.TechIdx.Length - 1
2768            Me.TechIdx(t) = srcProfile.TechIdx(t)
2769        Next t
2770        For v As Integer = 0 To srcProfile.VehIdx.Length - 1
2771            Me.VehIdx(v) = srcProfile.VehIdx(v)
2772        Next
2773        Me.RC = srcProfile.RC
2774    End Sub
2775    Public Sub AddVehicle(ByVal VIdx As Integer)
2776        If IsNothing(VehIdx) Then
2777            ReDim VehIdx(0)
2778            VehIdx(0) = VIdx
2779            RC = CAFE.Vehicles(VIdx).RegClass
2780        Else
2781            Dim j As Integer = VehIdx.Length
2782            ReDim Preserve VehIdx(j)
2783            VehIdx(j) = VIdx
2784        End If
2785    End Sub
2786
2787
2788    Public Shared Function GetLightTruckProfiles(ByRef Profiles() As TechProfile) As
        TechProfile()
2789        Dim OutProfiles() As TechProfile
2790        Dim o As Integer = 0
2791        For p As Integer = 0 To Profiles.Length - 1
2792            If Profiles(p).RC > 2 Then
2793                ReDim Preserve OutProfiles(o)
2794                OutProfiles(o) = Profiles(p)
2795                o += 1
2796            End If
2797        Next
2798        Return OutProfiles
2799    End Function
2800    Public Shared Function GetAutomobileProfiles(ByRef Profiles() As TechProfile) As
        TechProfile()
2801        Dim OutProfiles() As TechProfile
2802        Dim o As Integer = 0
2803        For p As Integer = 0 To Profiles.Length - 1
2804            If Profiles(p).RC = 1 Or Profiles(p).RC = 2 Then
2805                ReDim Preserve OutProfiles(o)
2806                OutProfiles(o) = Profiles(p)
2807                o += 1
2808            End If
2809        Next
2810        Return OutProfiles
2811    End Function
2812    Public Shared Function GetRCProfiles(ByRef Profiles() As TechProfile, ByVal WhichRC As
         Integer) As TechProfile()
2813        Dim OutProfiles() As TechProfile
2814        Dim o As Integer = 0
2815        For p As Integer = 0 To Profiles.Length - 1
2816            If Profiles(p).RC = WhichRC Then
2817                ReDim Preserve OutProfiles(o)
2818                OutProfiles(o) = Profiles(p)
2819                o += 1
2820            End If
2821        Next
2822        Return OutProfiles
```

```
2823    End Function
2824
2825    Public Sub AddTechnology(ByVal tIdx As Integer)
2826        If IsNothing(TechIdx) Then
2827            ReDim TechIdx(0)
2828            TechIdx(0) = tIdx
2829        Else
2830            Dim j As Integer = TechIdx.Length
2831            ReDim Preserve TechIdx(j)
2832            TechIdx(j) = tIdx
2833        End If
2834    End Sub
2835
2836    Public Shared Function IsDiffRCs(ByVal TechProf_A As TechProfile, _
2837                                     ByVal TechProf_B As TechProfile) As Boolean
2838        ' return true if the profiles involved have multiple RCs
2839        Dim tplist(1) As TechProfile
2840        tplist(0) = TechProf_A
2841        tplist(1) = TechProf_B
2842        Return TechProfile.IsDiffRCs(tplist)
2843    End Function
2844
2845    Public Shared Function IsDiffRCs(ByVal TechProfiles() As TechProfile) As Boolean
2846        If IsNothing(TechProfiles) OrElse TechProfiles.Length = 0 Then
2847            Return False
2848        End If
2849
2850        Dim LastRC As Integer = -1
2851        For Each tp As TechProfile In TechProfiles
2852            If (Not IsNothing(tp)) _
2853                    AndAlso (tp.RC > -1) Then
2854                If LastRC = -1 Then
2855                    LastRC = tp.RC
2856                Else
2857                    If tp.RC <> LastRC Then
2858                        Return True
2859                    End If
2860                End If
2861            End If
2862        Next tp
2863        ' if we get here, we didn't find any differences
2864        Return False
2865    End Function
2866    Public Shared Function IsAutoTruckMix(ByVal techprofiles() As TechProfile) As Boolean
2867        ' return True IF the profiles have both automobiles and light trucks
2868        Dim autofound As Boolean = False
2869        Dim truckfound As Boolean = False
2870        If IsNothing(techprofiles) OrElse techprofiles.Length = 0 Then
2871            Return False
2872        End If
2873
2874        Dim LastRC As Integer = -1
2875        For Each tp As TechProfile In techprofiles
2876            If (Not IsNothing(tp)) Then
2877                If (tp.RC = 1 Or tp.RC = 2) Then
2878                    autofound = True
2879                ElseIf tp.RC > 2 Then
2880                    truckfound = True
2881                End If
2882            End If
2883        Next tp
2884        ' if we get here, we didn't find any automobiles
2885        Return (autofound And truckfound)
2886    End Function
2887
2888    Public Function IsAuto() As Boolean
2889        Return Me._IsAuto
2890    End Function
2891    Public Function IsLightTruck() As Boolean
2892        Return Me._IsLightTruck
```

```
2893    End Function
2894    Public Function IsUnregulated() As Boolean
2895        Return Me._IsUnregulated
2896    End Function
2897
2898    Public Function CompareProfile(ByRef TechProf As TechProfile, _
2899                                   Optional ByRef MixedCarTruck As Boolean = False, _
2900                                   Optional ByRef MixedRCs As Boolean = False, _
2901                                   Optional ByRef MixedEnables As Boolean = False) As
        Boolean
2902        Return TechProfile.CompareProfile(TechProf, Me, MixedCarTruck, MixedRCs,
        MixedEnables)
2903    End Function
2904    Public Shared Function CompareProfile(ByRef TechProf_Orig As TechProfile, _
2905                                   ByRef TechProf_Sample As TechProfile, _
2906                                   Optional ByRef MixedCarTruck As Boolean = False, _
2907                                   Optional ByRef MixedRCs As Boolean = False, _
2908                                   Optional ByRef MixedEnables As Boolean = False) As
        Boolean
2909        Dim P() As TechProfile
2910        ReDim P(1)
2911        P(0) = TechProf_Orig
2912        P(1) = TechProf_Sample
2913        Return TechProfile.CompareProfile(P, MixedCarTruck, MixedRCs, MixedEnables)
2914    End Function
2915
2916    Public Shared Function CompareProfile(ByRef Profiles() As TechProfile, _
2917                        Optional ByRef MixedCarTruck As Boolean = False, _
2918                        Optional ByRef MixedRCs As Boolean = False, _
2919                        Optional ByRef MixedEnables As Boolean = False) As Boolean
2920        Dim SampleIsDifferent As Boolean = False
2921        Dim M_CarTruck As Boolean = False
2922        Dim M_RCs As Boolean = False
2923        Dim M_Enables As Boolean = False
2924
2925        ' Initial sanity clause: make sure there's at least two profiles to examine
2926        If IsNothing(Profiles) OrElse Profiles.Length < 2 Then
2927            Return False
2928        End If
2929
2930        ' Use the first profile as the 'original'
2931        Dim TechProf_Orig As TechProfile = Profiles(0)
2932
2933        If (IsNothing(TechProf_Orig)) OrElse _
2934            IsNothing(TechProf_Orig.VehIdx) OrElse _
2935            IsNothing(TechProf_Orig.TechIdx) OrElse _
2936            TechProf_Orig.VehIdx.Length = 0 OrElse _
2937            TechProf_Orig.TechIdx.Length = 0 Then
2938            MsgBox("Error with techprof_orig")        ' something wrong with "orig"...
        warn everyone
2939            Return False
2940        End If
2941        TechProf_Orig.Orig = True
2942        TechProf_Orig.DiffRC = False
2943        TechProf_Orig.DiffEnab = False
2944        TechProf_Orig.SetRCInfo()
2945
2946        For p As Integer = 1 To Profiles.Length - 1
2947
2948            Dim TechProf_Sample As TechProfile = Profiles(p)
2949
2950            If IsNothing(TechProf_Sample) = False Then
2951
2952                If (IsNothing(TechProf_Sample.VehIdx) OrElse TechProf_Sample.VehIdx.Length
        = 0) Or _
2953                    (IsNothing(TechProf_Orig.VehIdx) OrElse TechProf_Orig.VehIdx.Length =
        0) Or _
2954                    (IsNothing(TechProf_Sample.TechIdx) OrElse TechProf_Sample.TechIdx.
        Length = 0) Or _
2955                    (IsNothing(TechProf_Orig.TechIdx) OrElse TechProf_Orig.TechIdx.Length
```

```
= 0) Then
                    Return False    ' no vehicles or no tech in one or the other... let's split
                End If

                TechProf_Sample.Orig = False
                TechProf_Sample.DiffRC = False
                TechProf_Sample.DiffEnab = False
                TechProf_Sample.SetRCInfo()

                ' Check if the two profiles have the same RegClass
                If TechProf_Sample.RC <> TechProf_Orig.RC Then
                    TechProf_Sample.DiffRC = True
                    M_RCs = True
                    SampleIsDifferent = True
                    If TechProf_Sample.IsAuto = TechProf_Orig.IsLightTruck _
                        And (TechProf_Sample.IsAuto Or TechProf_Orig.IsAuto) Then
                        M_CarTruck = True
                    End If
                End If

                ' Check technologies
                If TechProf_Sample.TechIdx.Length <> TechProf_Orig.TechIdx.Length Then
                    TechProf_Sample.DiffEnab = True
                    M_Enables = True
                    SampleIsDifferent = True
                Else
                    Dim t As Integer
                    For t = 0 To TechProf_Sample.TechIdx.Length - 1
                        If TechProf_Sample.TechIdx(t) <> TechProf_Orig.TechIdx(t) Then
                            TechProf_Sample.DiffEnab = True
                            SampleIsDifferent = True
                        End If
                    Next t
                End If
            End If
        Next p

        ' All checks done ...
        If IsNothing(MixedCarTruck) = False Then
            MixedCarTruck = M_CarTruck
        End If
        If IsNothing(MixedRCs) = False Then
            MixedRCs = M_RCs
        End If
        If IsNothing(MixedEnables) = False Then
            MixedEnables = M_Enables
        End If
        Return Not SampleIsDifferent
    End Function

    Public Sub SetRCInfo()
        Me._IsAuto = False
        Me._IsLightTruck = False
        Me._IsUnregulated = False
        If (Me.RC = 1) Or (Me.RC) = 2 Then
            Me._IsAuto = True
        ElseIf Me.RC = 0 Then
            Me._IsUnregulated = True
        Else
            Me._IsLightTruck = True
        End If
    End Sub
    Public Sub AddComponent(ByVal ComponentType As Integer)
        NeedsNewComponent = True
        NewComponentType = ComponentType
    End Sub
End Class
#End Region
```

```
Imports System
Imports System.IO
Imports System.Collections
Module MyDEBUG

    ' the debug flag, used to determine whether to debug the model
    Public DBUG As Boolean = False

Debug Print Routines

DiagnosticFileBaseClass

DiagModelRunFile Class

DebugWriteActions Class

DiagModelErrorsFile Class

FinalTechTally Class

#Region "Summary Info Class"
    Public Class SummaryInfoClass
        Inherits DiagnosticFileClass

        Public Enum SummaryTypeEnum
            MfgFines = 1
            MfgAveragePriceIncr
            MfgTotalIncurredTechCost
            MfgAverageTechCost
            TotalIncurredTechCost
            TotalFines
            OverAllAveragePriceIncr
            OverAllAverageTechCost
            TotalLifeTimeFatalities
            LifeTimeFuelConsumption
            UndiscTotalFuelExpend
            UndiscFuelTaxRevenues
            UndiscTravelValue
            UndiscRefuelingTimeValue
            UndiscPetroMarketExternalities
            UndiscCongestionCosts
            UndiscNoiseCosts
            UndiscAccidentCosts
            UndiscCO2
            UndiscCO
            UndiscVOC
            UndiscNOX
            UndiscPM
            UndiscSOX
            DiscTotalFuelExpend
            DiscFuelTaxRevenues
            DiscTravelValue
            DiscRefuelingTimeValue
            DiscPetroMarketExternalities
            DiscCongestionCosts
            DiscNoiseCosts
            DiscAccidentCosts
            DiscCO2
            DiscCO
            DiscVOC
            DiscNOX
            DiscPM
            DiscSOX
            TotalSales
        End Enum
        Protected Structure MfgDataStructure
            Public Fines As Double
            Public AveragePriceIncrPerVehicle As Double
            Public AverageTechCost As Double
```

```
70              Public TotalIncurredTechCost As Double
71          End Structure
72          Protected Structure ScenarioDataStructure
73              Public MfgData() As MfgDataStructure
74              Public TotalTechOutlays As Double
75              Public TotalFines As Double
76              Public OverallAverageTechCost As Double
77              Public OverAllAveragePriceIncrPerVehicle As Double
78              Public TotalLifeTimeFatalities As Double
79              Public LifeTimeFuelConsumption As Double
80              Public UnDiscountedTotalLifeTimePreTaxFuelExpenditures As Double
81              Public UnDiscountedFuelTaxRevenues As Double
82              Public UnDiscountedTravelValue As Double
83              Public UnDiscountedRefuelingTimeValue As Double
84              Public UnDiscountedPetroMarketExternalities As Double
85              Public UnDiscountedCongestionCosts As Double
86              Public UnDiscountedNoiseCosts As Double
87              Public UnDiscountedAccidentCosts As Double
88              Public UnDiscountedCO2 As Double
89              Public UnDiscountedCO As Double
90              Public UnDiscountedVOC As Double
91              Public UnDiscountedNOX As Double
92              Public UnDiscountedPM As Double
93              Public UnDiscountedSOX As Double
94              Public DiscTotalLifetimePreTaxFuelExpenditures As Double
95              Public DiscountedFuelTaxRevenues As Double
96              Public DiscountedTravelValue As Double
97              Public DiscountedRefuelingTimeValue As Double
98              Public DiscountedPetroMarketExternalities As Double
99              Public DiscountedCongestionCosts As Double
100             Public DiscountedNoiseCosts As Double
101             Public DiscountedAccidentCosts As Double
102             Public DiscountedCO2 As Double
103             Public DiscountedCO As Double
104             Public DiscountedVOC As Double
105             Public DiscountedNOX As Double
106             Public DiscountedPM As Double
107             Public DiscountedSOX As Double
108             Public TotalSales As Double
109         End Structure
110         Protected Structure MYearDataStructure
111             Public MYearValid As Boolean
112             Public ScenarioData() As ScenarioDataStructure
113         End Structure
114         Protected SummaryInfo(-1) As MYearDataStructure
115         Public Sub New()
116
117             MyBase.New("Summary", -1, -1, -1, -1, False, False)
118
119         End Sub
120         Public Sub New(ByVal Iteration As Integer)
121             MyBase.New("Summary_" + Iteration.ToString, -1, -1, -1, -1, False, False)
122         End Sub
123
124         ' Write the compliance findings for this convergence loop...
125         Public Sub RecordSummary(ByVal SummaryType As SummaryTypeEnum, ByVal iScenario As
    Integer, ByVal MYear As Integer,
126                                  ByVal iMfg As Integer, ByVal SummaryData As Object)
127             ' see if we have space allocated for this MY
128             Dim iMY As Integer = CAFE.GMYI(MYear)
129             If SummaryInfo.Length = 0 Then
130                 ' first allocation...
131                 ' ALlocate new lengths and set them all invalid
132                 ReDim SummaryInfo(iMY)
133                 For i As Integer = 0 To iMY
134                     Me.SummaryInfo(i) = New MYearDataStructure
135                     Me.SummaryInfo(i).MYearValid = False
136                     ReDim Me.SummaryInfo(i).ScenarioData(-1)
137                 Next i
```

```
138             ElseIf iMY > SummaryInfo.Length - 1 Then
139                 Dim OldLength = SummaryInfo.Length
140                 ReDim Preserve SummaryInfo(iMY)
141                 SummaryInfo(iMY) = New MYearDataStructure
142                 ReDim Me.SummaryInfo(iMY).ScenarioData(-1)
143             End If
144
145             ' Now check out the number of scenarios within the MY structure
146             If iScenario > (SummaryInfo(iMY).ScenarioData.Length - 1) Then
147                 ReDim Preserve SummaryInfo(iMY).ScenarioData(iScenario)
148                 SummaryInfo(iMY).ScenarioData(iScenario) = New ScenarioDataStructure
149                 ReDim SummaryInfo(iMY).ScenarioData(iScenario).MfgData(-1)
150             End If
151
152             ' Finally, check if we're covered on the MFG data
153             If iMfg > (SummaryInfo(iMY).ScenarioData(iScenario).MfgData.Length - 1) Then
154                 ReDim Preserve SummaryInfo(iMY).ScenarioData(iScenario).MfgData(iMfg)
155                 SummaryInfo(iMY).ScenarioData(iScenario).MfgData(iMfg) = New MfgDataStructure
156             End If
157
158             ' Alright -- let's populate it
159             SummaryInfo(iMY).MYearValid = True
160             Select Case SummaryType
161                 Case SummaryTypeEnum.LifeTimeFuelConsumption
162                     SummaryInfo(iMY).ScenarioData(iScenario).LifeTimeFuelConsumption =
    SummaryData
163                 Case SummaryTypeEnum.MfgAveragePriceIncr
164                     SummaryInfo(iMY).ScenarioData(iScenario).MfgData(iMfg).
    AveragePriceIncrPerVehicle = SummaryData
165                 Case SummaryTypeEnum.MfgFines
166                     SummaryInfo(iMY).ScenarioData(iScenario).MfgData(iMfg).Fines =
    SummaryData
167                 Case SummaryTypeEnum.MfgTotalIncurredTechCost
168                     SummaryInfo(iMY).ScenarioData(iScenario).MfgData(iMfg).
    TotalIncurredTechCost = SummaryData
169                 Case SummaryTypeEnum.MfgAverageTechCost
170                     SummaryInfo(iMY).ScenarioData(iScenario).MfgData(iMfg).AverageTechCost =
    SummaryData
171                 Case SummaryTypeEnum.OverAllAveragePriceIncr
172                     SummaryInfo(iMY).ScenarioData(iScenario).
    OverAllAveragePriceIncrPerVehicle = SummaryData
173                 Case SummaryTypeEnum.OverAllAverageTechCost
174                     SummaryInfo(iMY).ScenarioData(iScenario).OverallAverageTechCost =
    SummaryData
175                 Case SummaryTypeEnum.TotalIncurredTechCost
176                     SummaryInfo(iMY).ScenarioData(iScenario).TotalTechOutlays = SummaryData
177                 Case SummaryTypeEnum.TotalFines
178                     SummaryInfo(iMY).ScenarioData(iScenario).TotalFines = SummaryData
179                 Case SummaryTypeEnum.TotalLifeTimeFatalities
180                     SummaryInfo(iMY).ScenarioData(iScenario).TotalLifeTimeFatalities =
    SummaryData
181                 Case SummaryTypeEnum.UndiscAccidentCosts
182                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedAccidentCosts =
    SummaryData
183                 Case SummaryTypeEnum.UndiscCO
184                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedCO = SummaryData
185                 Case SummaryTypeEnum.UndiscCO2
186                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedCO2 = SummaryData
187                 Case SummaryTypeEnum.UndiscCongestionCosts
188                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedCongestionCosts =
    SummaryData
189                 Case SummaryTypeEnum.UndiscFuelTaxRevenues
190                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedFuelTaxRevenues =
    SummaryData
191                 Case SummaryTypeEnum.UndiscNoiseCosts
192                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedNoiseCosts =
    SummaryData
193                 Case SummaryTypeEnum.UndiscNOX
194                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedNOX = SummaryData
```

```
195                 Case SummaryTypeEnum.UndiscPetroMarketExternalities
196                     SummaryInfo(iMY).ScenarioData(iScenario).
    UnDiscountedPetroMarketExternalities = SummaryData
197                 Case SummaryTypeEnum.UndiscPM
198                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedPM = SummaryData
199                 Case SummaryTypeEnum.UndiscRefuelingTimeValue
200                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedRefuelingTimeValue =
     SummaryData
201                 Case SummaryTypeEnum.UndiscSOX
202                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedSOX = SummaryData
203                 Case SummaryTypeEnum.UndiscTotalFuelExpend
204                     SummaryInfo(iMY).ScenarioData(iScenario).
    UnDiscountedTotalLifeTimePreTaxFuelExpenditures = SummaryData
205                 Case SummaryTypeEnum.UndiscTravelValue
206                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedTravelValue =
    SummaryData
207                 Case SummaryTypeEnum.UndiscVOC
208                     SummaryInfo(iMY).ScenarioData(iScenario).UnDiscountedVOC = SummaryData
209                 Case SummaryTypeEnum.DiscTotalFuelExpend
210                     SummaryInfo(iMY).ScenarioData(iScenario).
    DiscTotalLifetimePreTaxFuelExpenditures = SummaryData
211                 Case SummaryTypeEnum.DiscFuelTaxRevenues
212                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedFuelTaxRevenues =
    SummaryData
213                 Case SummaryTypeEnum.DiscAccidentCosts
214                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedAccidentCosts =
    SummaryData
215                 Case SummaryTypeEnum.DiscCO
216                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedCO = SummaryData
217                 Case SummaryTypeEnum.DiscCO2
218                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedCO2 = SummaryData
219                 Case SummaryTypeEnum.DiscCongestionCosts
220                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedCongestionCosts =
    SummaryData
221                 Case SummaryTypeEnum.DiscNoiseCosts
222                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedNoiseCosts =
    SummaryData
223                 Case SummaryTypeEnum.DiscNOX
224                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedNOX = SummaryData
225                 Case SummaryTypeEnum.DiscPetroMarketExternalities
226                     SummaryInfo(iMY).ScenarioData(iScenario).
    DiscountedPetroMarketExternalities = SummaryData
227                 Case SummaryTypeEnum.DiscPM
228                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedPM = SummaryData
229                 Case SummaryTypeEnum.DiscRefuelingTimeValue
230                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedRefuelingTimeValue =
    SummaryData
231                 Case SummaryTypeEnum.DiscSOX
232                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedSOX = SummaryData
233                 Case SummaryTypeEnum.DiscTravelValue
234                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedTravelValue =
    SummaryData
235                 Case SummaryTypeEnum.DiscVOC
236                     SummaryInfo(iMY).ScenarioData(iScenario).DiscountedVOC = SummaryData
237                 Case SummaryTypeEnum.TotalSales
238                     SummaryInfo(iMY).ScenarioData(iScenario).TotalSales = SummaryData
239                 Case Else
240                     Debug.WriteLine("Hey! Got summary type: " + SummaryType.ToString)
241             End Select
242
243         End Sub
244         Public Shadows Sub Close()
245             ' Write data to the file
246             Dim LineNo As Integer = 0
247             Dim imy As Integer
248             Dim isc As Integer
249             Dim imfg As Integer
250             Dim myStr As String
251             Dim scStr As String
```

```
252             Dim MfgStr As String
253             Dim OutHdr As String = String.Empty
254             Dim OutFatalities As String = String.Empty
255             Dim OutFuelConsumption As String = String.Empty
256             Dim OutUDFuelExpend As String = String.Empty
257             Dim OutUDFuelTaxRevenue As String = String.Empty
258             Dim OutUDTravelValue As String = String.Empty
259             Dim OutUDRefuelTimeValue As String = String.Empty
260             Dim OutUDPetroMktExternal As String = String.Empty
261             Dim OutUDCongestCost As String = String.Empty
262             Dim OutUDNoiseCost As String = String.Empty
263             Dim OutUDAccidentCost As String = String.Empty
264             Dim OutUDCO2 As String = String.Empty
265             Dim OutUDCO As String = String.Empty
266             Dim OutUDVOC As String = String.Empty
267             Dim OutUDNOX As String = String.Empty
268             Dim OutUDPM As String = String.Empty
269             Dim OutUDSOX As String = String.Empty
270             Dim OutDiscFuelExpend As String = String.Empty
271             Dim OutDiscFuelTaxRevenue As String = String.Empty
272             Dim OutDiscTravelValue As String = String.Empty
273             Dim OutDiscRefuelTimeValue As String = String.Empty
274             Dim OutDiscPetroMktExternal As String = String.Empty
275             Dim OutDiscCongestCost As String = String.Empty
276             Dim OutDiscNoiseCost As String = String.Empty
277             Dim OutDiscAccidentCost As String = String.Empty
278             Dim OutDiscCO2 As String = String.Empty
279             Dim OutDiscCO As String = String.Empty
280             Dim OutDiscVOC As String = String.Empty
281             Dim OutDiscNOX As String = String.Empty
282             Dim OutDiscPM As String = String.Empty
283             Dim OutDiscSOX As String = String.Empty
284             Dim IncrementalTechOutlays As Double = 0
285             Dim OutIncrementalTechOutlays As String = String.Empty
286             Dim DiscSocialBenefits As Double = 0
287             Dim OutDiscSocialBenefits As String = String.Empty
288             Dim OutBenCostRatio As String = String.Empty
289             Dim OutMfgTotalIncurredTechCost() As String
290             Dim OutTotalIncurredTechCost As String = String.Empty
291             Dim OutMfgFines() As String
292             Dim OutTotalFines As String = String.Empty
293             Dim OutMfgAveragePrice() As String
294             Dim OutOverallPrice As String = String.Empty
295             Dim OutMfgAverageTechCost() As String
296             Dim OutOverallAverageTechCost As String = String.Empty
297             Dim OutTotalSales As String = String.Empty
298
299             Dim MaxScenario As Integer = Integer.MinValue
300             Dim MinMYidx As Integer = Integer.MaxValue
301
302             ' Find the max scenario and the min (valid) MY
303             For imy = 0 To SummaryInfo.Length - 1
304                 If SummaryInfo(imy).MYearValid Then
305                     MinMYidx = Math.Min(MinMYidx, imy)
306                     MaxScenario = Math.Max(MaxScenario, SummaryInfo(imy).ScenarioData.Length)
307                 End If
308             Next imy
309             OutHdr = "n,MY,iMFG,MFG,Category"
310             LineNo = 0
311             For imy = MinMYidx To SummaryInfo.Length - 1
312                 myStr = "," + CAFE.GetMYearFromIndex(imy).ToString + ","
313                 '  Fill in row headers
314                 OutFatalities = LineNo.ToString + myStr + ",,Total Fatalities"
315                 LineNo += 1
316                 OutFuelConsumption = LineNo.ToString + myStr + ",,Total Fuel Consumption"
317                 LineNo += 1
318                 OutUDFuelExpend = LineNo.ToString + myStr + ",,Undisc Fuel Expenditures"
319                 LineNo += 1
320                 OutUDFuelTaxRevenue = LineNo.ToString + myStr + ",,Undisc Fuel Tax Revenue"
```

```
321                 LineNo += 1
322                 OutUDTravelValue = LineNo.ToString + myStr + ",,Undisc Travel Value"
323                 LineNo += 1
324                 OutUDRefuelTimeValue = LineNo.ToString + myStr + ",,Undisc Refuel Time Value"
325                 LineNo += 1
326                 OutUDPetroMktExternal = LineNo.ToString + myStr + ",,Undisc Petrol Mkt
    Externalities"
327                 LineNo += 1
328                 OutUDCongestCost = LineNo.ToString + myStr + ",,Undisc Congestion Cost"
329                 LineNo += 1
330                 OutUDNoiseCost = LineNo.ToString + myStr + ",,Undisc Noise Cost"
331                 LineNo += 1
332                 OutUDAccidentCost = LineNo.ToString + myStr + ",,Undisc Accident Cost"
333                 LineNo += 1
334                 OutUDCO2 = LineNo.ToString + myStr + ",,Undisc CO2"
335                 LineNo += 1
336                 OutUDCO = LineNo.ToString + myStr + ",,Undisc CO"
337                 LineNo += 1
338                 OutUDVOC = LineNo.ToString + myStr + ",,Undisc VOC"
339                 LineNo += 1
340                 OutUDNOX = LineNo.ToString + myStr + ",,Undisc NOX"
341                 LineNo += 1
342                 OutUDPM = LineNo.ToString + myStr + ",,Undisc PM"
343                 LineNo += 1
344                 OutUDSOX = LineNo.ToString + myStr + ",,Undisc SOX"
345                 LineNo += 1
346                 OutDiscFuelExpend = LineNo.ToString + myStr + ",,Disc Fuel Expenditures"
347                 LineNo += 1
348                 OutDiscFuelTaxRevenue = LineNo.ToString + myStr + ",,Disc Fuel Tax Revenue"
349                 LineNo += 1
350                 OutDiscTravelValue = LineNo.ToString + myStr + ",,Disc Travel Value"
351                 LineNo += 1
352                 OutDiscRefuelTimeValue = LineNo.ToString + myStr + ",,Disc Refuel Time Value"
353                 LineNo += 1
354                 OutDiscPetroMktExternal = LineNo.ToString + myStr + ",,Disc Petrol Mkt
    Externalities"
355                 LineNo += 1
356                 OutDiscCongestCost = LineNo.ToString + myStr + ",,Disc Congestion Cost"
357                 LineNo += 1
358                 OutDiscNoiseCost = LineNo.ToString + myStr + ",,Disc Noise Cost"
359                 LineNo += 1
360                 OutDiscAccidentCost = LineNo.ToString + myStr + ",,Disc Accident Cost"
361                 LineNo += 1
362                 OutDiscCO2 = LineNo.ToString + myStr + ",,Disc CO2"
363                 LineNo += 1
364                 OutDiscCO = LineNo.ToString + myStr + ",,Disc CO"
365                 LineNo += 1
366                 OutDiscVOC = LineNo.ToString + myStr + ",,Disc VOC"
367                 LineNo += 1
368                 OutDiscNOX = LineNo.ToString + myStr + ",,Disc NOX"
369                 LineNo += 1
370                 OutDiscPM = LineNo.ToString + myStr + ",,Disc PM"
371                 LineNo += 1
372                 OutDiscSOX = LineNo.ToString + myStr + ",,Disc SOX"
373                 LineNo += 1
374                 OutDiscSocialBenefits = LineNo.ToString + myStr + ",,Social Benefits (NPV)"
375                 LineNo += 1
376                 OutIncrementalTechOutlays = LineNo.ToString + myStr + ",,Incremental Tech.
    Outlays"
377                 LineNo += 1
378                 OutBenCostRatio = LineNo.ToString + myStr + ",,Benefit:Cost Ratio"
379                 LineNo += 1
380                 ' Assume that the number of MFGs remains the same for al scenarios in all MYs
381                 ' and that they match the mfg indexes in CAFE.mfgs
382                 ReDim OutMfgTotalIncurredTechCost(CAFE.Mfgs.Length - 1)
383                 For imfg = 0 To CAFE.Mfgs.Length - 1
384                     OutMfgTotalIncurredTechCost(imfg) = LineNo.ToString + myStr + imfg.
    ToString + "," + CAFE.Mfgs(imfg).MfgName + ",Total Incurred Tech Cost"
385                     LineNo += 1
```

```
386                 Next
387                 OutTotalIncurredTechCost = LineNo.ToString + myStr + ",,Total Incurred Tech
    Cost"
388                 LineNo += 1
389                 ReDim OutMfgFines(CAFE.Mfgs.Length - 1)
390                 For imfg = 0 To CAFE.Mfgs.Length - 1
391                     OutMfgFines(imfg) = LineNo.ToString + myStr + imfg.ToString + "," + CAFE.
    Mfgs(imfg).MfgName + ",Fines"
392                     LineNo += 1
393                 Next
394                 OutTotalFines = LineNo.ToString + myStr + ",,Total Fines"
395                 LineNo += 1
396                 ReDim OutMfgAverageTechCost(CAFE.Mfgs.Length - 1)
397                 For imfg = 0 To CAFE.Mfgs.Length - 1
398                     OutMfgAverageTechCost(imfg) = LineNo.ToString + myStr + imfg.ToString + "
    ," + CAFE.Mfgs(imfg).MfgName + ",Average Tech Cost"
399                     LineNo += 1
400                 Next
401                 OutOverallAverageTechCost = LineNo.ToString + myStr + ",,Overall Average Tech
    Cost"
402                 ReDim OutMfgAveragePrice(CAFE.Mfgs.Length - 1)
403                 For imfg = 0 To CAFE.Mfgs.Length - 1
404                     OutMfgAveragePrice(imfg) = LineNo.ToString + myStr + imfg.ToString + ","
    + CAFE.Mfgs(imfg).MfgName + ",Average Price"
405                     LineNo += 1
406                 Next
407                 OutOverallPrice = LineNo.ToString + myStr + ",,Overall Price Increase"
408                 LineNo += 1
409                 OutTotalSales = LineNo.ToString + myStr + ",,Total Sales"
410                 LineNo += 1
411
412                 ' Now fill in the scenario info, going across the page
413                 For isc = 0 To MaxScenario - 1
414                     OutHdr += ",Scenario " + isc.ToString
415                     OutFatalities += "," + SummaryInfo(imy).ScenarioData(isc).
    TotalLifeTimeFatalities.ToString
416                     OutFuelConsumption += "," + SummaryInfo(imy).ScenarioData(isc).
    LifeTimeFuelConsumption.ToString
417                     OutUDFuelExpend += "," + SummaryInfo(imy).ScenarioData(isc).
    UnDiscountedTotalLifeTimePreTaxFuelExpenditures.ToString
418                     OutUDFuelTaxRevenue += "," + SummaryInfo(imy).ScenarioData(isc).
    UnDiscountedFuelTaxRevenues.ToString
419                     OutUDTravelValue += "," + SummaryInfo(imy).ScenarioData(isc).
    UnDiscountedTravelValue.ToString
420                     OutUDRefuelTimeValue += "," + SummaryInfo(imy).ScenarioData(isc).
    UnDiscountedRefuelingTimeValue.ToString
421                     OutUDPetroMktExternal += "," + SummaryInfo(imy).ScenarioData(isc).
    UnDiscountedPetroMarketExternalities.ToString
422                     OutUDCongestCost += "," + SummaryInfo(imy).ScenarioData(isc).
    UnDiscountedCongestionCosts.ToString
423                     OutUDNoiseCost += "," + SummaryInfo(imy).ScenarioData(isc).
    UnDiscountedNoiseCosts.ToString
424                     OutUDAccidentCost += "," + SummaryInfo(imy).ScenarioData(isc).
    UnDiscountedAccidentCosts.ToString
425                     OutUDCO2 += "," + SummaryInfo(imy).ScenarioData(isc).UnDiscountedCO2.
    ToString
426                     OutUDCO += "," + SummaryInfo(imy).ScenarioData(isc).UnDiscountedCO.
    ToString
427                     OutUDVOC += "," + SummaryInfo(imy).ScenarioData(isc).UnDiscountedVOC.
    ToString
428                     OutUDNOX += "," + SummaryInfo(imy).ScenarioData(isc).UnDiscountedNOX.
    ToString
429                     OutUDPM += "," + SummaryInfo(imy).ScenarioData(isc).UnDiscountedPM.
    ToString
430                     OutUDSOX += "," + SummaryInfo(imy).ScenarioData(isc).UnDiscountedSOX.
    ToString
431                     OutDiscFuelExpend += "," + SummaryInfo(imy).ScenarioData(isc).
    DiscTotalLifetimePreTaxFuelExpenditures.ToString
432                     OutDiscFuelTaxRevenue += "," + SummaryInfo(imy).ScenarioData(isc).
```

```
DiscountedFuelTaxRevenues.ToString
                    OutDiscTravelValue += "," + SummaryInfo(imy).ScenarioData(isc).
DiscountedTravelValue.ToString
                    OutDiscRefuelTimeValue += "," + SummaryInfo(imy).ScenarioData(isc).
DiscountedRefuelingTimeValue.ToString
                    OutDiscPetroMktExternal += "," + SummaryInfo(imy).ScenarioData(isc).
DiscountedPetroMarketExternalities.ToString
                    OutDiscCongestCost += "," + SummaryInfo(imy).ScenarioData(isc).
DiscountedCongestionCosts.ToString
                    OutDiscNoiseCost += "," + SummaryInfo(imy).ScenarioData(isc).
DiscountedNoiseCosts.ToString
                    OutDiscAccidentCost += "," + SummaryInfo(imy).ScenarioData(isc).
DiscountedAccidentCosts.ToString
                    OutDiscCO2 += "," + SummaryInfo(imy).ScenarioData(isc).DiscountedCO2.
ToString
                    OutDiscCO += "," + SummaryInfo(imy).ScenarioData(isc).DiscountedCO.
ToString
                    OutDiscVOC += "," + SummaryInfo(imy).ScenarioData(isc).DiscountedVOC.
ToString
                    OutDiscNOX += "," + SummaryInfo(imy).ScenarioData(isc).DiscountedNOX.
ToString
                    OutDiscPM += "," + SummaryInfo(imy).ScenarioData(isc).DiscountedPM.
ToString
                    OutDiscSOX += "," + SummaryInfo(imy).ScenarioData(isc).DiscountedSOX.
ToString
                    If isc = 0 Then
                        DiscSocialBenefits = 0
                        IncrementalTechOutlays = 0
                        OutBenCostRatio += ","
                    Else
                        DiscSocialBenefits = (SummaryInfo(imy).ScenarioData(0).
DiscTotalLifetimePreTaxFuelExpenditures _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedTravelValue _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedRefuelingTimeValue _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedPetroMarketExternalities
_
                        + SummaryInfo(imy).ScenarioData(0).DiscountedCongestionCosts _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedNoiseCosts _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedAccidentCosts _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedCO2 _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedCO _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedVOC _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedNOX _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedPM _
                        + SummaryInfo(imy).ScenarioData(0).DiscountedSOX) _
                        - (SummaryInfo(imy).ScenarioData(isc).
DiscTotalLifetimePreTaxFuelExpenditures _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedTravelValue _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedRefuelingTimeValue _
                        + SummaryInfo(imy).ScenarioData(isc).
DiscountedPetroMarketExternalities _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedCongestionCosts _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedNoiseCosts _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedAccidentCosts _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedCO2 _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedCO _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedVOC _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedNOX _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedPM _
                        + SummaryInfo(imy).ScenarioData(isc).DiscountedSOX)
                        IncrementalTechOutlays = SummaryInfo(imy).ScenarioData(isc).
TotalTechOutlays - _
                        SummaryInfo(imy).ScenarioData(0).TotalTechOutlays
                        If IncrementalTechOutlays <> 0 Then
                            OutBenCostRatio += "," + (DiscSocialBenefits / (1000 *
IncrementalTechOutlays)).ToString
                        Else
                            OutBenCostRatio += ","
                        End If
```

```
483                     End If
484                     OutDiscSocialBenefits += "," + DiscSocialBenefits.ToString
485                     OutIncrementalTechOutlays += "," + IncrementalTechOutlays.ToString
486
487                     For imfg = 0 To CAFE.Mfgs.Length - 1
488                         OutMfgTotalIncurredTechCost(imfg) += "," + SummaryInfo(imy).
    ScenarioData(isc).MfgData(imfg).TotalIncurredTechCost.ToString
489                     OutMfgFines(imfg) += "," + SummaryInfo(imy).ScenarioData(isc).MfgData
    (imfg).Fines.ToString
490                     OutMfgAverageTechCost(imfg) += "," + SummaryInfo(imy).ScenarioData
    (isc).MfgData(imfg).AverageTechCost.ToString
491                     OutMfgAveragePrice(imfg) += "," + SummaryInfo(imy).ScenarioData(isc).
    MfgData(imfg).AveragePriceIncrPerVehicle.ToString
492                     Next
493                     OutTotalIncurredTechCost += "," + SummaryInfo(imy).ScenarioData(isc).
    TotalTechOutlays.ToString
494                     OutTotalFines += "," + SummaryInfo(imy).ScenarioData(isc).TotalFines.
    ToString
495                     OutOverallAverageTechCost += "," + SummaryInfo(imy).ScenarioData(isc).
    OverallAverageTechCost.ToString
496                     OutOverallPrice += "," + SummaryInfo(imy).ScenarioData(isc).
    OverAllAveragePriceIncrPerVehicle.ToString
497                     OutTotalSales += "," + SummaryInfo(imy).ScenarioData(isc).TotalSales.
    ToString
498                 Next isc
499                 ' Now output the info for all scenarios for this MY.
500                 ' If it's the first MY, then output the column headers.
501                 If imy = MinMYidx Then
502                     OutputFile.WriteLine(OutHdr)
503                 End If
504                 OutputFile.WriteLine(OutFatalities)
505                 OutputFile.WriteLine(OutFuelConsumption)
506                 OutputFile.WriteLine(OutUDFuelExpend)
507                 OutputFile.WriteLine(OutUDFuelTaxRevenue)
508                 OutputFile.WriteLine(OutUDTravelValue)
509                 OutputFile.WriteLine(OutUDRefuelTimeValue)
510                 OutputFile.WriteLine(OutUDPetroMktExternal)
511                 OutputFile.WriteLine(OutUDCongestCost)
512                 OutputFile.WriteLine(OutUDNoiseCost)
513                 OutputFile.WriteLine(OutUDAccidentCost)
514                 OutputFile.WriteLine(OutUDCO2)
515                 OutputFile.WriteLine(OutUDCO)
516                 OutputFile.WriteLine(OutUDVOC)
517                 OutputFile.WriteLine(OutUDNOX)
518                 OutputFile.WriteLine(OutUDPM)
519                 OutputFile.WriteLine(OutUDSOX)
520                 OutputFile.WriteLine(OutDiscFuelExpend)
521                 OutputFile.WriteLine(OutDiscFuelTaxRevenue)
522                 OutputFile.WriteLine(OutDiscTravelValue)
523                 OutputFile.WriteLine(OutDiscRefuelTimeValue)
524                 OutputFile.WriteLine(OutDiscPetroMktExternal)
525                 OutputFile.WriteLine(OutDiscCongestCost)
526                 OutputFile.WriteLine(OutDiscNoiseCost)
527                 OutputFile.WriteLine(OutDiscAccidentCost)
528                 OutputFile.WriteLine(OutDiscCO2)
529                 OutputFile.WriteLine(OutDiscCO)
530                 OutputFile.WriteLine(OutDiscVOC)
531                 OutputFile.WriteLine(OutDiscNOX)
532                 OutputFile.WriteLine(OutDiscPM)
533                 OutputFile.WriteLine(OutDiscSOX)
534                 OutputFile.WriteLine(OutDiscSocialBenefits)
535                 OutputFile.WriteLine(OutIncrementalTechOutlays)
536                 OutputFile.WriteLine(OutBenCostRatio)
537                 For imfg = 0 To CAFE.Mfgs.Length - 1
538                     OutputFile.WriteLine(OutMfgTotalIncurredTechCost(imfg))
539                 Next
540                 OutputFile.WriteLine(OutTotalIncurredTechCost)
541                 For imfg = 0 To CAFE.Mfgs.Length - 1
542                     OutputFile.WriteLine(OutMfgFines(imfg))
```

```
543                 Next
544                 OutputFile.WriteLine(OutTotalFines)
545                 For imfg = 0 To CAFE.Mfgs.Length - 1
546                     OutputFile.WriteLine(OutMfgAverageTechCost(imfg))
547                 Next
548                 OutputFile.WriteLine(OutOverallAverageTechCost)
549                 For imfg = 0 To CAFE.Mfgs.Length - 1
550                     OutputFile.WriteLine(OutMfgAveragePrice(imfg))
551                 Next
552                 OutputFile.WriteLine(OutOverallPrice)
553                 OutputFile.WriteLine(OutTotalSales)
554             Next imy
555             MyBase.Close()
556         End Sub
557     End Class
558 #End Region
559
560 End Module
561
```

```
Public Class MYForm
    Inherits System.Windows.Forms.Form

    ' Create CheckBox control array to track model year checkboxes
    Private MYControlArray As CheckboxArrayClass
    Private MYParent As Object
    ' Define the form's look and feel
    Private MYCheckBoxOrientation As Integer    ' define checkbox control array orientation

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        Me.init()

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents m_CancelButton As System.Windows.Forms.Button
    Friend WithEvents m_OKButton As System.Windows.Forms.Button
    Public WithEvents m_GroupBoxMY As System.Windows.Forms.GroupBox
    Friend WithEvents m_CheckAll As System.Windows.Forms.Button
    Friend WithEvents m_UnCheck As System.Windows.Forms.Button
    Friend WithEvents m_ChangeOrientation As System.Windows.Forms.Button
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Me.m_CancelButton = New System.Windows.Forms.Button
        Me.m_GroupBoxMY = New System.Windows.Forms.GroupBox
        Me.m_ChangeOrientation = New System.Windows.Forms.Button
        Me.m_OKButton = New System.Windows.Forms.Button
        Me.m_CheckAll = New System.Windows.Forms.Button
        Me.m_UnCheck = New System.Windows.Forms.Button
        Me.m_GroupBoxMY.SuspendLayout()
        Me.SuspendLayout()
        '
        'm_CancelButton
        '
        Me.m_CancelButton.DialogResult = System.Windows.Forms.DialogResult.Cancel
        Me.m_CancelButton.Location = New System.Drawing.Point(336, 200)
        Me.m_CancelButton.Name = "m_CancelButton"
        Me.m_CancelButton.Size = New System.Drawing.Size(96, 32)
        Me.m_CancelButton.TabIndex = 4
        Me.m_CancelButton.Tag = ""
        Me.m_CancelButton.Text = "&Cancel"
        '
        'm_GroupBoxMY
        '
        Me.m_GroupBoxMY.Controls.Add(Me.m_ChangeOrientation)
        Me.m_GroupBoxMY.Location = New System.Drawing.Point(16, 16)
        Me.m_GroupBoxMY.Name = "m_GroupBoxMY"
        Me.m_GroupBoxMY.Size = New System.Drawing.Size(424, 176)
```

```
71          Me.m_GroupBoxMY.TabIndex = 0
72          Me.m_GroupBoxMY.TabStop = False
73          Me.m_GroupBoxMY.Text = "Model Years"
74          '
75          'm_ChangeOrientation
76          '
77          Me.m_ChangeOrientation.FlatStyle = System.Windows.Forms.FlatStyle.Flat
78          Me.m_ChangeOrientation.Font = New System.Drawing.Font("Arial", 6.75!, System.
        Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
79          Me.m_ChangeOrientation.Location = New System.Drawing.Point(413, 7)
80          Me.m_ChangeOrientation.Name = "m_ChangeOrientation"
81          Me.m_ChangeOrientation.Size = New System.Drawing.Size(8, 8)
82          Me.m_ChangeOrientation.TabIndex = 99
83          Me.m_ChangeOrientation.TabStop = False
84          Me.m_ChangeOrientation.Text = "o"
85          '
86          'm_OKButton
87          '
88          Me.m_OKButton.BackColor = System.Drawing.SystemColors.Control
89          Me.m_OKButton.Location = New System.Drawing.Point(232, 200)
90          Me.m_OKButton.Name = "m_OKButton"
91          Me.m_OKButton.Size = New System.Drawing.Size(96, 32)
92          Me.m_OKButton.TabIndex = 3
93          Me.m_OKButton.Text = "&OK"
94          '
95          'm_CheckAll
96          '
97          Me.m_CheckAll.BackColor = System.Drawing.SystemColors.Control
98          Me.m_CheckAll.Location = New System.Drawing.Point(24, 200)
99          Me.m_CheckAll.Name = "m_CheckAll"
100         Me.m_CheckAll.Size = New System.Drawing.Size(96, 32)
101         Me.m_CheckAll.TabIndex = 1
102         Me.m_CheckAll.Text = "Check &All"
103         '
104         'm_UnCheck
105         '
106         Me.m_UnCheck.Location = New System.Drawing.Point(128, 200)
107         Me.m_UnCheck.Name = "m_UnCheck"
108         Me.m_UnCheck.Size = New System.Drawing.Size(96, 32)
109         Me.m_UnCheck.TabIndex = 2
110         Me.m_UnCheck.Text = "&Uncheck All"
111         '
112         'MYForm
113         '
114         Me.AcceptButton = Me.m_OKButton
115         Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
116         Me.CancelButton = Me.m_CancelButton
117         Me.ClientSize = New System.Drawing.Size(456, 245)
118         Me.ControlBox = False
119         Me.Controls.Add(Me.m_OKButton)
120         Me.Controls.Add(Me.m_CheckAll)
121         Me.Controls.Add(Me.m_GroupBoxMY)
122         Me.Controls.Add(Me.m_CancelButton)
123         Me.Controls.Add(Me.m_UnCheck)
124         Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D
125         Me.MaximizeBox = False
126         Me.MinimizeBox = False
127         Me.Name = "MYForm"
128         Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent
129         Me.Text = "Model Years"
130         Me.m_GroupBoxMY.ResumeLayout(False)
131         Me.ResumeLayout(False)
132
133     End Sub
134
135 #End Region
136
137     Public Sub New(ByVal Sender As Object, _
138                    ByVal CheckBoxOrientation As Integer)
139         MyBase.New()
```

```
140
141         'This call is required by the Windows Form Designer.
142         InitializeComponent()
143
144         Me.MYParent = Sender
145         Me.MYCheckBoxOrientation = CheckBoxOrientation
146
147         init()
148     End Sub
149
150     Public Sub init()
151         MYControlArray = New CheckboxArrayClass(m_GroupBoxMY)       ' Initialize MY control array
152
153         Dim i As Integer
154         'Dim ModelYearsAvailable() As Object = CAFE.ModelYearsAvailable.ToArray
155         Dim MinYear As Integer = 2000
156         Dim MaxYear As Integer = CAFE.GetMYearFromIndex(CAFE.ModelYearsSelected.Length - 1)
157         Dim ModelYearsAvailable(MaxYear - MinYear) As Integer
158
159         For i = 0 To ModelYearsAvailable.Length - 1
160             ModelYearsAvailable(i) = MinYear + i
161         Next i
162
163         For i = 0 To ModelYearsAvailable.Length - 1
164             ' Console.WriteLine(ModelYearsAvailable(i))
165             MYControlArray.AddNewCheckbox((ModelYearsAvailable(i)).ToString)
166         Next i
167
168         MYControlArray.SetOrientation(MYCheckBoxOrientation)
169     End Sub
170
171     Private Sub m_CheckAll_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles m_CheckAll.Click
172         Dim i As Integer
173
174         For i = 0 To MYControlArray.Count - 1
175             MYControlArray(i).Checked = True
176         Next i
177     End Sub
178
179     Private Sub m_UnCheck_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles m_UnCheck.Click
180         Dim i As Integer
181
182         For i = 0 To MYControlArray.Count - 1
183             MYControlArray(i).Checked = False
184         Next i
185     End Sub
186
187     Private Sub m_OKButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles m_OKButton.Click
188         Dim i As Integer
189         'ReDim Preserve CAFE.ModelYearsSelected(MYControlArray.Count)
190
191         For i = 0 To MYControlArray.Count - 1
192             CAFE.ModelYearsSelected(i) = MYControlArray(i).Checked
193         Next i
194
195         'DirectCast(MYParent, CafeForm).SetMYLookAndFeel(Me.MYCheckBoxOrientation)
196
197         Me.Close()
198     End Sub
199
200     Private Sub m_CancelButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles m_CancelButton.Click
201         Me.Close()
202     End Sub
203
204     Private Sub MYForm_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

```
    Handles MyBase.Load
205         Dim i As Integer
206         Dim MYToolTip As New System.Windows.Forms.ToolTip
207
208         For i = 0 To MYControlArray.Count - 1
209             MYControlArray(i).Checked = CAFE.ModelYearsSelected(i)
210             ' Set tooltips for year checkboxes
211             MYToolTip.SetToolTip(MYControlArray(i), " Check-on to model " & _
212                         "data for year " & MYControlArray(i).Tag & ". ")
213         Next i
214
215         ' Set tooltip information
216         MYToolTip.SetToolTip(m_CheckAll, " Click to select all available years. ")
217         MYToolTip.SetToolTip(m_UnCheck, " Click to deselect all available years. ")
218         MYToolTip.SetToolTip(m_OKButton, " Click OK to save the changes. ")
219         MYToolTip.SetToolTip(m_CancelButton, " Click Cancel to disregard the changes. ")
220
221         Me.Show()
222
223         If MYControlArray.RowsColsExceeded Then
224             MessageBox.Show("Note: this form contains more information " & _
225                             "that can be displayed on the screen.", "Model Years", _
226                             MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
227         End If
228     End Sub
229
230     Private Sub m_ChangeOrientation_Click(ByVal sender As System.Object, ByVal e As System. ↙
        EventArgs) Handles m_ChangeOrientation.Click
231         MYControlArray.ChangeOrientation()
232         MYCheckBoxOrientation = MYControlArray.GetOrientation()
233     End Sub
234
235     Private Sub m_GroupBoxMY_Enter(ByVal sender As System.Object, ByVal e As System.      ↙
        EventArgs) Handles m_GroupBoxMY.Enter
236
237     End Sub
238 End Class
239
```

```
1  Public Class NamePlateClass
2
3      Public Valid As Boolean                ' Run-time generated
4      Public MyIndexNumber As Integer        ' Run-time generated, stores the index of this nameplate
5      Public MfgIdx As Integer               ' Mfg Index for this Mfg.
6
7      Public Name As String           ' Stores the name of this nameplate
8      Public nCompleteModels As Integer       ' number of entries in CompleteModelsList
9      Public CompleteModelsList() As Integer  ' Complete list of models which use this nameplate
10     Public nModels As Integer               ' Number of valid models that use this nameplate
11     Public ModelsList() As Integer          ' List of valid models that use this nameplate
12     Public TechCost As Double               ' Tech cost associated with this nameplate.
13     Public TechImprv As Double              ' Tech improvement in FE
14     Public TechUsed() As Boolean            ' Nameplate techs which have been used go here
15
16     Public Sub New()
17         Me.Valid = False
18         Me.nModels = 0
19         Me.ModelsList = Nothing
20         Me.TechCost = 0
21         Me.TechImprv = 1
22         ReDim Me.TechUsed(-1)
23     End Sub
24
25     Public Sub Copy(ByRef Dst As NamePlateClass)
26         NamePlateClass.Copy(Dst, Me)
27     End Sub
28
29     Public Shared Sub Copy(ByRef Dst As NamePlateClass, ByRef Src As NamePlateClass)
30         Dim i As Integer
31
32         Dst = New NamePlateClass
33
34         Dst.Valid = Src.Valid
35         Dst.MyIndexNumber = Src.MyIndexNumber
36         Dst.MfgIdx = Src.MfgIdx
37
38         Dst.Name = Src.Name
39
40         Dst.nCompleteModels = Src.nCompleteModels
41         ReDim Dst.CompleteModelsList(Dst.nCompleteModels - 1)
42         For i = 0 To Dst.nCompleteModels - 1
43             Dst.CompleteModelsList(i) = Src.CompleteModelsList(i)
44         Next
45
46         Dst.nModels = Src.nModels
47         If Src.nModels > 0 Then
48             ReDim Dst.ModelsList(Src.nModels - 1)
49             For i = 0 To Src.nModels - 1
50                 Dst.ModelsList(i) = Src.ModelsList(i)
51             Next
52         End If
53
54         Dst.TechCost = Src.TechCost
55         Dst.TechImprv = Src.TechImprv
56         If IsNothing(Src.TechUsed) Then
57             ReDim Dst.TechUsed(Src.TechUsed.Length - 1)
58             For i = 0 To Src.TechUsed.Length - 1
59                 Dst.TechUsed(i) = Src.TechUsed(i)
60             Next
61         End If
62     End Sub
63
64     'Public Sub AddUser(ByVal VehIndex As Integer)
65     '    ' Add a vehicle (index to vehicles) as a user which uses this nameplate
66     '    ReDim Preserve ModelsList(nModels)
67     '    ModelsList(nModels) = VehIndex
```

```
68      '    nModels = nModels + 1
69      'End Sub
70      Public Sub AddUser(ByVal VehIndex As Integer)
71          ' Add a vehicle (index to vehicles) as a vehicle which uses this engine
72          If IsNothing(CompleteModelsList) Then
73              ReDim CompleteModelsList(-1)
74          End If
75          ReDim Preserve CompleteModelsList(nCompleteModels)
76          CompleteModelsList(nCompleteModels) = VehIndex
77          nCompleteModels = nCompleteModels + 1
78
79      End Sub
80
81      '''
82      ' CreateModelsList
83      ' This sub populates the ModelsList from valid vehicles in the CompleteModelsList
84      '
85      Public Sub CreateModelsList()
86          nModels = 0
87          ReDim ModelsList(-1)
88          For v As Integer = 0 To CompleteModelsList.Length - 1
89              If CAFE.Vehicles(CompleteModelsList(v)).IsValid Then
90                  ' got a valid one
91                  ReDim Preserve ModelsList(ModelsList.Length)
92                  ModelsList(ModelsList.Length - 1) = CompleteModelsList(v)
93              End If
94          Next
95          nModels = ModelsList.Length
96          Dim NumTechnologies = CAFE.Technologies.GetNumTechnologies
97      End Sub
98
99  End Class
100
```

```
Public Class OptionsForm
    Inherits System.Windows.Forms.Form

    ' stores whether the options have been changed since the form was last loaded
    Private ChangeDetected As Boolean

    ' define options selections
    Private PathSettings As CAFE.PathsEnum
    Private FuelConsumptionSavingsEstimates As CAFE.LowAverageHigh
    Private TechnologyCostEstimates As CAFE.LowAverageHigh

    Private MyToolTip As System.Windows.Forms.ToolTip

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents m_GroupBoxPath1 As System.Windows.Forms.RadioButton
    Friend WithEvents m_GroupBoxPath2 As System.Windows.Forms.RadioButton
    Friend WithEvents m_GroupBoxPath3 As System.Windows.Forms.RadioButton
    Friend WithEvents m_GroupBoxFuelLow As System.Windows.Forms.RadioButton
    Friend WithEvents m_GroupBoxFuelAverage As System.Windows.Forms.RadioButton
    Friend WithEvents m_GroupBoxFuelHigh As System.Windows.Forms.RadioButton
    Friend WithEvents m_GroupBoxCostLow As System.Windows.Forms.RadioButton
    Friend WithEvents m_GroupBoxCostAverage As System.Windows.Forms.RadioButton
    Friend WithEvents m_ModelYearsButton As System.Windows.Forms.Button
    Friend WithEvents m_OKButton As System.Windows.Forms.Button
    Friend WithEvents m_CancelButton As System.Windows.Forms.Button
    Friend WithEvents m_ApplyButton As System.Windows.Forms.Button
    Friend WithEvents m_OptionsTabControl As System.Windows.Forms.TabControl
    Friend WithEvents m_TechSettingsTabPage As System.Windows.Forms.TabPage
    Friend WithEvents m_OperatingModesTabPage As System.Windows.Forms.TabPage
    Friend WithEvents m_OtherTabPage As System.Windows.Forms.TabPage
    Friend WithEvents m_NoFinesModeCheckBox As System.Windows.Forms.CheckBox
    Friend WithEvents m_LowCostTechnologyFirstModeCheckBox As System.Windows.Forms.
    CheckBox
    Friend WithEvents m_BackFillTechnologyModeCheckBox As System.Windows.Forms.CheckBox
    Friend WithEvents m_CreateTransmissionIfCarsInvolvedModeCheckBox As System.Windows.
    Forms.CheckBox
    Friend WithEvents m_CreateTransmissionIfMultipleRCModeCheckBox As System.Windows.Forms
    .CheckBox
    Friend WithEvents m_CreateTransmissionIfTechEnableMissMatchModeCheckBox As System.
    Windows.Forms.CheckBox
    Friend WithEvents m_CreateEngineIfCarsInvolvedModeCheckBox As System.Windows.Forms.
    CheckBox
    Friend WithEvents m_CreateEngineIfMultipleRCModeCheckBox As System.Windows.Forms.
```

```
    CheckBox
66      Friend WithEvents m_CreateEngineIfTechEnableMissMatchModeCheckBox As System.Windows.
        Forms.CheckBox
67      Friend WithEvents m_OperatingModesGroupBox As System.Windows.Forms.GroupBox
68      Friend WithEvents m_SelectAllModesCheckBox As System.Windows.Forms.CheckBox
69      Friend WithEvents m_EncryptionSettingsTabPage As System.Windows.Forms.TabPage
70      Friend WithEvents m_PathsGroupBox As System.Windows.Forms.GroupBox
71      Friend WithEvents m_FuelGroupBox As System.Windows.Forms.GroupBox
72      Friend WithEvents m_CostGroupBox As System.Windows.Forms.GroupBox
73      Friend WithEvents m_GroupBoxCostHigh As System.Windows.Forms.RadioButton
74      Friend WithEvents m_EncryptionSettingsSavingGroupBox As System.Windows.Forms.GroupBox
75      Friend WithEvents m_EncryptionSettingsLoadingGroupBox As System.Windows.Forms.GroupBox
76      Friend WithEvents m_EncryptOutputFilesCheckBox As System.Windows.Forms.CheckBox
77      Friend WithEvents m_EncryptionProviderComboBox As System.Windows.Forms.ComboBox
78      Friend WithEvents m_EncryptionProviderLabel As System.Windows.Forms.Label
79      Friend WithEvents m_KeyLengthLabel As System.Windows.Forms.Label
80      Friend WithEvents m_KeyLengthNumericUpDown As System.Windows.Forms.NumericUpDown
81      Friend WithEvents m_EncryptDocPropertiesCheckBox As System.Windows.Forms.CheckBox
82      Friend WithEvents m_EncryptOutputFilesPanel As System.Windows.Forms.Panel
83      Friend WithEvents m_ReadEncryptedFilesPanel As System.Windows.Forms.Panel
84      Friend WithEvents m_InputSinglePwTextBox As System.Windows.Forms.TextBox
85      Friend WithEvents m_InputMultiPwCheckBox As System.Windows.Forms.CheckBox
86      Friend WithEvents m_InputSinglePwCheckBox As System.Windows.Forms.CheckBox
87      Friend WithEvents m_ReadEncryptedFilesCheckBox As System.Windows.Forms.CheckBox
88      Friend WithEvents m_InputMultiPwVehLabel As System.Windows.Forms.Label
89      Friend WithEvents m_InputMultiPwVehTextBox As System.Windows.Forms.TextBox
90      Friend WithEvents m_InputMultiPwEngLabel As System.Windows.Forms.Label
91      Friend WithEvents m_InputMultiPwEngTextBox As System.Windows.Forms.TextBox
92      Friend WithEvents m_InputMultiPwTranTextBox As System.Windows.Forms.TextBox
93      Friend WithEvents m_InputMultiPwTranLabel As System.Windows.Forms.Label
94      Friend WithEvents m_InputMultiPwTechTextBox As System.Windows.Forms.TextBox
95      Friend WithEvents m_InputMultiPwTechLabel As System.Windows.Forms.Label
96      Friend WithEvents m_PasswordToModifyTextBox As System.Windows.Forms.TextBox
97      Friend WithEvents m_PasswordToModifyLabel As System.Windows.Forms.Label
98      Friend WithEvents m_PasswordToOpenTextBox As System.Windows.Forms.TextBox
99      Friend WithEvents m_PasswordToOpenLabel As System.Windows.Forms.Label
100     Friend WithEvents m_PasswordToModifyCheckBox As System.Windows.Forms.CheckBox
101     Friend WithEvents m_FolderBrowserDialog As System.Windows.Forms.FolderBrowserDialog
102     Friend WithEvents m_RunEffectsModelCheckBox As System.Windows.Forms.CheckBox
103     Friend WithEvents m_OutputSettingsTabPage As System.Windows.Forms.TabPage
104     Friend WithEvents m_GeneralGroupBox As System.Windows.Forms.GroupBox
105     Friend WithEvents m_DefaultOutputPathTextBox As System.Windows.Forms.TextBox
106     Friend WithEvents m_DefaultOutputPathLabel As System.Windows.Forms.Label
107     Friend WithEvents m_DefaultOutputPathBrowseButton As System.Windows.Forms.Button
108     Friend WithEvents m_HideUnusedRCsCheckBox As System.Windows.Forms.CheckBox
109     Friend WithEvents m_ReportSettingsGroupBox As System.Windows.Forms.GroupBox
110     Friend WithEvents m_GenIndustryReportCheckBox As System.Windows.Forms.CheckBox
111     Friend WithEvents m_GenEffectsReportCheckBox As System.Windows.Forms.CheckBox
112     Friend WithEvents m_GenAllReportsCheckBox As System.Windows.Forms.CheckBox
113     Friend WithEvents m_GenVehiclesReportCheckBox As System.Windows.Forms.CheckBox
114     Friend WithEvents m_GenMfgsReportCheckBox As System.Windows.Forms.CheckBox
115     Friend WithEvents m_OtherSettingsGroupBox As System.Windows.Forms.GroupBox
116     Friend WithEvents m_NoStartupWarningCheckBox As System.Windows.Forms.CheckBox
117     Friend WithEvents m_AutoInvokeEfficiencyModeCheckBox As System.Windows.Forms.CheckBox
118     Friend WithEvents m_SingleFileInputCheckBox As System.Windows.Forms.CheckBox
119     Friend WithEvents m_ThreeFileInputCheckBox As System.Windows.Forms.CheckBox
120     Friend WithEvents m_UseOutputTemplatesCheckBox As System.Windows.Forms.CheckBox
121     Friend WithEvents m_ConfigureTemplatesButton As System.Windows.Forms.Button
122     Friend WithEvents m_OutputTemplatesGroupBox As System.Windows.Forms.GroupBox
123     'Friend WithEvents m_TemplatesTabPage As System.Windows.Forms.TabPage
124     Friend WithEvents m_TemplateSettingsGroupBox As System.Windows.Forms.GroupBox
125     Friend WithEvents m_UseOutputTemplatesCheckBox2 As System.Windows.Forms.CheckBox
126     Friend WithEvents m_UseOutputTemplatesPanel As System.Windows.Forms.Panel
127     Friend WithEvents m_DefaultOutputTemplateLabel As System.Windows.Forms.Label
128     Friend WithEvents m_LandscapeOutputTemplateLabel As System.Windows.Forms.Label
129     Friend WithEvents m_LandscapeOutputTemplateTextBox As System.Windows.Forms.TextBox
130     Friend WithEvents m_LandscapeOutputTemplateBrowseButton As System.Windows.Forms.Button
131     Friend WithEvents m_PortraitOutputTemplateLabel As System.Windows.Forms.Label
132     Friend WithEvents m_PortraitOutputTemplateTextBox As System.Windows.Forms.TextBox
133     Friend WithEvents m_PortraitOutputTemplateBrowseButton As System.Windows.Forms.Button
```

```
134     Friend WithEvents m_OtherTemplatesGroupBox As System.Windows.Forms.GroupBox
135     Friend WithEvents Label1 As System.Windows.Forms.Label
136     Friend WithEvents m_OpenFileDialog As System.Windows.Forms.OpenFileDialog
137     Friend WithEvents m_GenEAReportCheckBox As System.Windows.Forms.CheckBox
138     Friend WithEvents m_StageAnalysisCheckBox As System.Windows.Forms.CheckBox
139     Friend WithEvents m_DefaultOutputTemplateBrowseButton2 As System.Windows.Forms.Button
140     Friend WithEvents m_DefaultOutputTemplateTextBox2 As System.Windows.Forms.TextBox
141     Friend WithEvents m_DefaultOutputTemplateBrowseButton As System.Windows.Forms.Button
142     Friend WithEvents m_DefaultOutputTemplateTextBox As System.Windows.Forms.TextBox
143     Friend WithEvents m_EnableProfilingCheckBox As System.Windows.Forms.CheckBox
144     Friend WithEvents m_EnableProfilingMemoryCheckBox As System.Windows.Forms.CheckBox
145     Friend WithEvents m_MonteCarloAnalysisCheckBox As System.Windows.Forms.CheckBox
146     Friend WithEvents m_MonteCarloIterationsLabel As System.Windows.Forms.Label
147     Friend WithEvents m_MonteCarloIterationsNumericUpDown As System.Windows.Forms.
        NumericUpDown
148     Friend WithEvents m_ReserveTechsIfInComplianceModeCheckBox As System.Windows.Forms.
        CheckBox
149     Friend WithEvents m_CreateEngineIfLabel As System.Windows.Forms.Label
150     Friend WithEvents m_CreateTransmissionIfLabel As System.Windows.Forms.Label
151     Friend WithEvents m_SkipEngineTransmissionIfLabel As System.Windows.Forms.Label
152     Friend WithEvents m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox As System.
        Windows.Forms.CheckBox
153     Friend WithEvents m_SkipEngineTransmissionIfMultipleRCModeCheckBox As System.Windows.
        Forms.CheckBox
154     Friend WithEvents m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox As System.Windows
        .Forms.CheckBox
155     Friend WithEvents m_CreateEngineIfAllModesCheckBox As System.Windows.Forms.CheckBox
156     Friend WithEvents m_CreateTransmissionIfAllModesCheckBox As System.Windows.Forms.
        CheckBox
157     Friend WithEvents m_SkipEngineTransmissionIfAllModesCheckBox As System.Windows.Forms.
        CheckBox
158     <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
159         Me.m_PathsGroupBox = New System.Windows.Forms.GroupBox
160         Me.m_GroupBoxPath3 = New System.Windows.Forms.RadioButton
161         Me.m_GroupBoxPath2 = New System.Windows.Forms.RadioButton
162         Me.m_GroupBoxPath1 = New System.Windows.Forms.RadioButton
163         Me.m_FuelGroupBox = New System.Windows.Forms.GroupBox
164         Me.m_GroupBoxFuelHigh = New System.Windows.Forms.RadioButton
165         Me.m_GroupBoxFuelAverage = New System.Windows.Forms.RadioButton
166         Me.m_GroupBoxFuelLow = New System.Windows.Forms.RadioButton
167         Me.m_CostGroupBox = New System.Windows.Forms.GroupBox
168         Me.m_GroupBoxCostHigh = New System.Windows.Forms.RadioButton
169         Me.m_GroupBoxCostAverage = New System.Windows.Forms.RadioButton
170         Me.m_GroupBoxCostLow = New System.Windows.Forms.RadioButton
171         Me.m_ModelYearsButton = New System.Windows.Forms.Button
172         Me.m_OKButton = New System.Windows.Forms.Button
173         Me.m_CancelButton = New System.Windows.Forms.Button
174         Me.m_ApplyButton = New System.Windows.Forms.Button
175         Me.m_OptionsTabControl = New System.Windows.Forms.TabControl
176         Me.m_TechSettingsTabPage = New System.Windows.Forms.TabPage
177         Me.m_OperatingModesTabPage = New System.Windows.Forms.TabPage
178         Me.m_OperatingModesGroupBox = New System.Windows.Forms.GroupBox
179         Me.m_SkipEngineTransmissionIfAllModesCheckBox = New System.Windows.Forms.CheckBox
180         Me.m_CreateTransmissionIfAllModesCheckBox = New System.Windows.Forms.CheckBox
181         Me.m_CreateEngineIfAllModesCheckBox = New System.Windows.Forms.CheckBox
182         Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox = New System.Windows.
        Forms.CheckBox
183         Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox = New System.Windows.Forms.
        CheckBox
184         Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox = New System.Windows.Forms.
        CheckBox
185         Me.m_SkipEngineTransmissionIfLabel = New System.Windows.Forms.Label
186         Me.m_MonteCarloIterationsNumericUpDown = New System.Windows.Forms.NumericUpDown
187         Me.m_MonteCarloIterationsLabel = New System.Windows.Forms.Label
188         Me.m_MonteCarloAnalysisCheckBox = New System.Windows.Forms.CheckBox
189         Me.m_ReserveTechsIfInComplianceModeCheckBox = New System.Windows.Forms.CheckBox
190         Me.m_StageAnalysisCheckBox = New System.Windows.Forms.CheckBox
191         Me.m_AutoInvokeEfficiencyModeCheckBox = New System.Windows.Forms.CheckBox
192         Me.m_SelectAllModesCheckBox = New System.Windows.Forms.CheckBox
193         Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox = New System.Windows.
```

```
    Forms.CheckBox
194         Me.m_CreateTransmissionIfMultipleRCModeCheckBox = New System.Windows.Forms.
    CheckBox
195         Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox = New System.Windows.Forms.
    CheckBox
196         Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox = New System.Windows.Forms.
    CheckBox
197         Me.m_CreateEngineIfMultipleRCModeCheckBox = New System.Windows.Forms.CheckBox
198         Me.m_CreateEngineIfCarsInvolvedModeCheckBox = New System.Windows.Forms.CheckBox
199         Me.m_BackFillTechnologyModeCheckBox = New System.Windows.Forms.CheckBox
200         Me.m_LowCostTechnologyFirstModeCheckBox = New System.Windows.Forms.CheckBox
201         Me.m_NoFinesModeCheckBox = New System.Windows.Forms.CheckBox
202         Me.m_CreateEngineIfLabel = New System.Windows.Forms.Label
203         Me.m_CreateTransmissionIfLabel = New System.Windows.Forms.Label
204         Me.m_EncryptionSettingsTabPage = New System.Windows.Forms.TabPage
205         Me.m_EncryptionSettingsSavingGroupBox = New System.Windows.Forms.GroupBox
206         Me.m_EncryptOutputFilesPanel = New System.Windows.Forms.Panel
207         Me.m_EncryptDocPropertiesCheckBox = New System.Windows.Forms.CheckBox
208         Me.m_PasswordToModifyTextBox = New System.Windows.Forms.TextBox
209         Me.m_PasswordToModifyLabel = New System.Windows.Forms.Label
210         Me.m_KeyLengthLabel = New System.Windows.Forms.Label
211         Me.m_EncryptionProviderLabel = New System.Windows.Forms.Label
212         Me.m_PasswordToOpenTextBox = New System.Windows.Forms.TextBox
213         Me.m_PasswordToModifyCheckBox = New System.Windows.Forms.CheckBox
214         Me.m_EncryptionProviderComboBox = New System.Windows.Forms.ComboBox
215         Me.m_KeyLengthNumericUpDown = New System.Windows.Forms.NumericUpDown
216         Me.m_PasswordToOpenLabel = New System.Windows.Forms.Label
217         Me.m_EncryptOutputFilesCheckBox = New System.Windows.Forms.CheckBox
218         Me.m_EncryptionSettingsLoadingGroupBox = New System.Windows.Forms.GroupBox
219         Me.m_ReadEncryptedFilesPanel = New System.Windows.Forms.Panel
220         Me.m_InputMultiPwTechTextBox = New System.Windows.Forms.TextBox
221         Me.m_InputMultiPwTechLabel = New System.Windows.Forms.Label
222         Me.m_InputMultiPwTranTextBox = New System.Windows.Forms.TextBox
223         Me.m_InputMultiPwTranLabel = New System.Windows.Forms.Label
224         Me.m_InputMultiPwEngTextBox = New System.Windows.Forms.TextBox
225         Me.m_InputMultiPwEngLabel = New System.Windows.Forms.Label
226         Me.m_InputMultiPwVehTextBox = New System.Windows.Forms.TextBox
227         Me.m_InputMultiPwVehLabel = New System.Windows.Forms.Label
228         Me.m_InputSinglePwTextBox = New System.Windows.Forms.TextBox
229         Me.m_InputMultiPwCheckBox = New System.Windows.Forms.CheckBox
230         Me.m_InputSinglePwCheckBox = New System.Windows.Forms.CheckBox
231         Me.m_ReadEncryptedFilesCheckBox = New System.Windows.Forms.CheckBox
232         Me.m_OutputSettingsTabPage = New System.Windows.Forms.TabPage
233         Me.m_TemplateSettingsGroupBox = New System.Windows.Forms.GroupBox
234         Me.m_DefaultOutputTemplateBrowseButton = New System.Windows.Forms.Button
235         Me.m_DefaultOutputTemplateTextBox = New System.Windows.Forms.TextBox
236         Me.m_ConfigureTemplatesButton = New System.Windows.Forms.Button
237         Me.m_UseOutputTemplatesCheckBox = New System.Windows.Forms.CheckBox
238         Me.m_ReportSettingsGroupBox = New System.Windows.Forms.GroupBox
239         Me.m_GenEAReportCheckBox = New System.Windows.Forms.CheckBox
240         Me.m_GenAllReportsCheckBox = New System.Windows.Forms.CheckBox
241         Me.m_GenVehiclesReportCheckBox = New System.Windows.Forms.CheckBox
242         Me.m_GenMfgsReportCheckBox = New System.Windows.Forms.CheckBox
243         Me.m_GenIndustryReportCheckBox = New System.Windows.Forms.CheckBox
244         Me.m_GenEffectsReportCheckBox = New System.Windows.Forms.CheckBox
245         Me.m_GeneralGroupBox = New System.Windows.Forms.GroupBox
246         Me.m_DefaultOutputPathTextBox = New System.Windows.Forms.TextBox
247         Me.m_DefaultOutputPathLabel = New System.Windows.Forms.Label
248         Me.m_DefaultOutputPathBrowseButton = New System.Windows.Forms.Button
249         Me.m_HideUnusedRCsCheckBox = New System.Windows.Forms.CheckBox
250         Me.m_OtherTabPage = New System.Windows.Forms.TabPage
251         Me.m_OtherSettingsGroupBox = New System.Windows.Forms.GroupBox
252         Me.m_EnableProfilingMemoryCheckBox = New System.Windows.Forms.CheckBox
253         Me.m_EnableProfilingCheckBox = New System.Windows.Forms.CheckBox
254         Me.m_ThreeFileInputCheckBox = New System.Windows.Forms.CheckBox
255         Me.m_SingleFileInputCheckBox = New System.Windows.Forms.CheckBox
256         Me.m_RunEffectsModelCheckBox = New System.Windows.Forms.CheckBox
257         Me.m_NoStartupWarningCheckBox = New System.Windows.Forms.CheckBox
258         Me.m_OtherTemplatesGroupBox = New System.Windows.Forms.GroupBox
259         Me.Label1 = New System.Windows.Forms.Label
```

```
260         Me.m_OutputTemplatesGroupBox = New System.Windows.Forms.GroupBox
261         Me.m_UseOutputTemplatesPanel = New System.Windows.Forms.Panel
262         Me.m_PortraitOutputTemplateBrowseButton = New System.Windows.Forms.Button
263         Me.m_PortraitOutputTemplateTextBox = New System.Windows.Forms.TextBox
264         Me.m_PortraitOutputTemplateLabel = New System.Windows.Forms.Label
265         Me.m_LandscapeOutputTemplateBrowseButton = New System.Windows.Forms.Button
266         Me.m_LandscapeOutputTemplateTextBox = New System.Windows.Forms.TextBox
267         Me.m_LandscapeOutputTemplateLabel = New System.Windows.Forms.Label
268         Me.m_DefaultOutputTemplateBrowseButton2 = New System.Windows.Forms.Button
269         Me.m_DefaultOutputTemplateTextBox2 = New System.Windows.Forms.TextBox
270         Me.m_DefaultOutputTemplateLabel = New System.Windows.Forms.Label
271         Me.m_UseOutputTemplatesCheckBox2 = New System.Windows.Forms.CheckBox
272         Me.m_FolderBrowserDialog = New System.Windows.Forms.FolderBrowserDialog
273         Me.m_OpenFileDialog = New System.Windows.Forms.OpenFileDialog
274         Me.m_PathsGroupBox.SuspendLayout()
275         Me.m_FuelGroupBox.SuspendLayout()
276         Me.m_CostGroupBox.SuspendLayout()
277         Me.m_OptionsTabControl.SuspendLayout()
278         Me.m_TechSettingsTabPage.SuspendLayout()
279         Me.m_OperatingModesTabPage.SuspendLayout()
280         Me.m_OperatingModesGroupBox.SuspendLayout()
281         CType(Me.m_MonteCarloIterationsNumericUpDown, System.ComponentModel.
        ISupportInitialize).BeginInit()
282         Me.m_EncryptionSettingsTabPage.SuspendLayout()
283         Me.m_EncryptionSettingsSavingGroupBox.SuspendLayout()
284         Me.m_EncryptOutputFilesPanel.SuspendLayout()
285         CType(Me.m_KeyLengthNumericUpDown, System.ComponentModel.ISupportInitialize).
        BeginInit()
286         Me.m_EncryptionSettingsLoadingGroupBox.SuspendLayout()
287         Me.m_ReadEncryptedFilesPanel.SuspendLayout()
288         Me.m_OutputSettingsTabPage.SuspendLayout()
289         Me.m_TemplateSettingsGroupBox.SuspendLayout()
290         Me.m_ReportSettingsGroupBox.SuspendLayout()
291         Me.m_GeneralGroupBox.SuspendLayout()
292         Me.m_OtherTabPage.SuspendLayout()
293         Me.m_OtherSettingsGroupBox.SuspendLayout()
294         Me.m_OtherTemplatesGroupBox.SuspendLayout()
295         Me.m_OutputTemplatesGroupBox.SuspendLayout()
296         Me.m_UseOutputTemplatesPanel.SuspendLayout()
297         Me.SuspendLayout()
298         '
299         'm_PathsGroupBox
300         '
301         Me.m_PathsGroupBox.Controls.Add(Me.m_GroupBoxPath3)
302         Me.m_PathsGroupBox.Controls.Add(Me.m_GroupBoxPath2)
303         Me.m_PathsGroupBox.Controls.Add(Me.m_GroupBoxPath1)
304         Me.m_PathsGroupBox.Location = New System.Drawing.Point(8, 8)
305         Me.m_PathsGroupBox.Name = "m_PathsGroupBox"
306         Me.m_PathsGroupBox.Size = New System.Drawing.Size(616, 56)
307         Me.m_PathsGroupBox.TabIndex = 0
308         Me.m_PathsGroupBox.TabStop = False
309         Me.m_PathsGroupBox.Text = "Technology Paths"
310         '
311         'm_GroupBoxPath3
312         '
313         Me.m_GroupBoxPath3.Location = New System.Drawing.Point(352, 24)
314         Me.m_GroupBoxPath3.Name = "m_GroupBoxPath3"
315         Me.m_GroupBoxPath3.Size = New System.Drawing.Size(64, 16)
316         Me.m_GroupBoxPath3.TabIndex = 2
317         Me.m_GroupBoxPath3.Text = "Path &3"
318         '
319         'm_GroupBoxPath2
320         '
321         Me.m_GroupBoxPath2.Location = New System.Drawing.Point(184, 24)
322         Me.m_GroupBoxPath2.Name = "m_GroupBoxPath2"
323         Me.m_GroupBoxPath2.Size = New System.Drawing.Size(64, 16)
324         Me.m_GroupBoxPath2.TabIndex = 1
325         Me.m_GroupBoxPath2.Text = "Path &2"
326         '
327         'm_GroupBoxPath1
```

```
328         '
329         Me.m_GroupBoxPath1.Location = New System.Drawing.Point(16, 24)
330         Me.m_GroupBoxPath1.Name = "m_GroupBoxPath1"
331         Me.m_GroupBoxPath1.Size = New System.Drawing.Size(64, 16)
332         Me.m_GroupBoxPath1.TabIndex = 0
333         Me.m_GroupBoxPath1.Text = "Path &1"
334         '
335         'm_FuelGroupBox
336         '
337         Me.m_FuelGroupBox.Controls.Add(Me.m_GroupBoxFuelHigh)
338         Me.m_FuelGroupBox.Controls.Add(Me.m_GroupBoxFuelAverage)
339         Me.m_FuelGroupBox.Controls.Add(Me.m_GroupBoxFuelLow)
340         Me.m_FuelGroupBox.Location = New System.Drawing.Point(8, 72)
341         Me.m_FuelGroupBox.Name = "m_FuelGroupBox"
342         Me.m_FuelGroupBox.Size = New System.Drawing.Size(616, 56)
343         Me.m_FuelGroupBox.TabIndex = 1
344         Me.m_FuelGroupBox.TabStop = False
345         Me.m_FuelGroupBox.Text = "Technology Fuel Consumption Estimates"
346         '
347         'm_GroupBoxFuelHigh
348         '
349         Me.m_GroupBoxFuelHigh.Location = New System.Drawing.Point(352, 24)
350         Me.m_GroupBoxFuelHigh.Name = "m_GroupBoxFuelHigh"
351         Me.m_GroupBoxFuelHigh.Size = New System.Drawing.Size(136, 16)
352         Me.m_GroupBoxFuelHigh.TabIndex = 2
353         Me.m_GroupBoxFuelHigh.Text = "Use &High Estimates"
354         '
355         'm_GroupBoxFuelAverage
356         '
357         Me.m_GroupBoxFuelAverage.Location = New System.Drawing.Point(184, 24)
358         Me.m_GroupBoxFuelAverage.Name = "m_GroupBoxFuelAverage"
359         Me.m_GroupBoxFuelAverage.Size = New System.Drawing.Size(152, 16)
360         Me.m_GroupBoxFuelAverage.TabIndex = 1
361         Me.m_GroupBoxFuelAverage.Text = "Use &Average Estimates"
362         '
363         'm_GroupBoxFuelLow
364         '
365         Me.m_GroupBoxFuelLow.Location = New System.Drawing.Point(16, 24)
366         Me.m_GroupBoxFuelLow.Name = "m_GroupBoxFuelLow"
367         Me.m_GroupBoxFuelLow.Size = New System.Drawing.Size(128, 16)
368         Me.m_GroupBoxFuelLow.TabIndex = 0
369         Me.m_GroupBoxFuelLow.Text = "Use &Low Estimates"
370         '
371         'm_CostGroupBox
372         '
373         Me.m_CostGroupBox.Controls.Add(Me.m_GroupBoxCostHigh)
374         Me.m_CostGroupBox.Controls.Add(Me.m_GroupBoxCostAverage)
375         Me.m_CostGroupBox.Controls.Add(Me.m_GroupBoxCostLow)
376         Me.m_CostGroupBox.Location = New System.Drawing.Point(8, 136)
377         Me.m_CostGroupBox.Name = "m_CostGroupBox"
378         Me.m_CostGroupBox.Size = New System.Drawing.Size(616, 56)
379         Me.m_CostGroupBox.TabIndex = 2
380         Me.m_CostGroupBox.TabStop = False
381         Me.m_CostGroupBox.Text = "Technology Cost Estimates"
382         '
383         'm_GroupBoxCostHigh
384         '
385         Me.m_GroupBoxCostHigh.Location = New System.Drawing.Point(352, 24)
386         Me.m_GroupBoxCostHigh.Name = "m_GroupBoxCostHigh"
387         Me.m_GroupBoxCostHigh.Size = New System.Drawing.Size(152, 16)
388         Me.m_GroupBoxCostHigh.TabIndex = 2
389         Me.m_GroupBoxCostHigh.Text = "Use &High Cost Estimates"
390         '
391         'm_GroupBoxCostAverage
392         '
393         Me.m_GroupBoxCostAverage.Location = New System.Drawing.Point(184, 24)
394         Me.m_GroupBoxCostAverage.Name = "m_GroupBoxCostAverage"
395         Me.m_GroupBoxCostAverage.Size = New System.Drawing.Size(176, 16)
396         Me.m_GroupBoxCostAverage.TabIndex = 1
397         Me.m_GroupBoxCostAverage.Text = "Use &Average Cost Estimates"
```

```
398         '
399         'm_GroupBoxCostLow
400         '
401         Me.m_GroupBoxCostLow.Location = New System.Drawing.Point(16, 24)
402         Me.m_GroupBoxCostLow.Name = "m_GroupBoxCostLow"
403         Me.m_GroupBoxCostLow.Size = New System.Drawing.Size(152, 16)
404         Me.m_GroupBoxCostLow.TabIndex = 0
405         Me.m_GroupBoxCostLow.Text = "Use &Low Cost Estimates"
406         '
407         'm_ModelYearsButton
408         '
409         Me.m_ModelYearsButton.Anchor = System.Windows.Forms.AnchorStyles.Bottom
410         Me.m_ModelYearsButton.Location = New System.Drawing.Point(111, 280)
411         Me.m_ModelYearsButton.Name = "m_ModelYearsButton"
412         Me.m_ModelYearsButton.Size = New System.Drawing.Size(96, 32)
413         Me.m_ModelYearsButton.TabIndex = 101
414         Me.m_ModelYearsButton.Text = "&Model Years"
415         '
416         'm_OKButton
417         '
418         Me.m_OKButton.Anchor = System.Windows.Forms.AnchorStyles.Bottom
419         Me.m_OKButton.Location = New System.Drawing.Point(319, 280)
420         Me.m_OKButton.Name = "m_OKButton"
421         Me.m_OKButton.Size = New System.Drawing.Size(96, 32)
422         Me.m_OKButton.TabIndex = 103
423         Me.m_OKButton.Text = "&OK"
424         '
425         'm_CancelButton
426         '
427         Me.m_CancelButton.Anchor = System.Windows.Forms.AnchorStyles.Bottom
428         Me.m_CancelButton.Location = New System.Drawing.Point(423, 280)
429         Me.m_CancelButton.Name = "m_CancelButton"
430         Me.m_CancelButton.Size = New System.Drawing.Size(96, 32)
431         Me.m_CancelButton.TabIndex = 104
432         Me.m_CancelButton.Text = "&Cancel"
433         '
434         'm_ApplyButton
435         '
436         Me.m_ApplyButton.Anchor = System.Windows.Forms.AnchorStyles.Bottom
437         Me.m_ApplyButton.Location = New System.Drawing.Point(215, 280)
438         Me.m_ApplyButton.Name = "m_ApplyButton"
439         Me.m_ApplyButton.Size = New System.Drawing.Size(96, 32)
440         Me.m_ApplyButton.TabIndex = 102
441         Me.m_ApplyButton.Text = "&Apply"
442         '
443         'm_OptionsTabControl
444         '
445         Me.m_OptionsTabControl.Controls.Add(Me.m_TechSettingsTabPage)
446         Me.m_OptionsTabControl.Controls.Add(Me.m_OperatingModesTabPage)
447         Me.m_OptionsTabControl.Controls.Add(Me.m_EncryptionSettingsTabPage)
448         Me.m_OptionsTabControl.Controls.Add(Me.m_OutputSettingsTabPage)
449         Me.m_OptionsTabControl.Controls.Add(Me.m_OtherTabPage)
450         Me.m_OptionsTabControl.Dock = System.Windows.Forms.DockStyle.Top
451         Me.m_OptionsTabControl.Location = New System.Drawing.Point(0, 0)
452         Me.m_OptionsTabControl.Name = "m_OptionsTabControl"
453         Me.m_OptionsTabControl.SelectedIndex = 0
454         Me.m_OptionsTabControl.Size = New System.Drawing.Size(638, 264)
455         Me.m_OptionsTabControl.TabIndex = 8
456         '
457         'm_TechSettingsTabPage
458         '
459         Me.m_TechSettingsTabPage.Controls.Add(Me.m_FuelGroupBox)
460         Me.m_TechSettingsTabPage.Controls.Add(Me.m_PathsGroupBox)
461         Me.m_TechSettingsTabPage.Controls.Add(Me.m_CostGroupBox)
462         Me.m_TechSettingsTabPage.Location = New System.Drawing.Point(4, 22)
463         Me.m_TechSettingsTabPage.Name = "m_TechSettingsTabPage"
464         Me.m_TechSettingsTabPage.Size = New System.Drawing.Size(630, 238)
465         Me.m_TechSettingsTabPage.TabIndex = 0
466         Me.m_TechSettingsTabPage.Text = "Technology Settings"
467         '
```

```
468         'm_OperatingModesTabPage
469         '
470         Me.m_OperatingModesTabPage.Controls.Add(Me.m_OperatingModesGroupBox)
471         Me.m_OperatingModesTabPage.Location = New System.Drawing.Point(4, 22)
472         Me.m_OperatingModesTabPage.Name = "m_OperatingModesTabPage"
473         Me.m_OperatingModesTabPage.Size = New System.Drawing.Size(630, 238)
474         Me.m_OperatingModesTabPage.TabIndex = 1
475         Me.m_OperatingModesTabPage.Text = "Operating Modes"
476         '
477         'm_OperatingModesGroupBox
478         '
479         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_SkipEngineTransmissionIfAllModesCheckBox)
480         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_CreateTransmissionIfAllModesCheckBox)
481         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_CreateEngineIfAllModesCheckBox)
482         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox)
483         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_SkipEngineTransmissionIfMultipleRCModeCheckBox)
484         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox)
485         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_SkipEngineTransmissionIfLabel)
486         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_MonteCarloIterationsNumericUpDown)
487         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_MonteCarloIterationsLabel)
488         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_MonteCarloAnalysisCheckBox)
489         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_ReserveTechsIfInComplianceModeCheckBox)
490         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_StageAnalysisCheckBox)
491         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_AutoInvokeEfficiencyModeCheckBox)
492         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_SelectAllModesCheckBox)
493         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_CreateTransmissionIfTechEnableMissMatchModeCheckBox)
494         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_CreateTransmissionIfMultipleRCModeCheckBox)
495         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_CreateTransmissionIfCarsInvolvedModeCheckBox)
496         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_CreateEngineIfTechEnableMissMatchModeCheckBox)
497         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_CreateEngineIfMultipleRCModeCheckBox)
498         Me.m_OperatingModesGroupBox.Controls.Add(Me.
        m_CreateEngineIfCarsInvolvedModeCheckBox)
499         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_BackFillTechnologyModeCheckBox)
500         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_LowCostTechnologyFirstModeCheckBox)
501         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_NoFinesModeCheckBox)
502         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_CreateEngineIfLabel)
503         Me.m_OperatingModesGroupBox.Controls.Add(Me.m_CreateTransmissionIfLabel)
504         Me.m_OperatingModesGroupBox.Location = New System.Drawing.Point(8, 8)
505         Me.m_OperatingModesGroupBox.Name = "m_OperatingModesGroupBox"
506         Me.m_OperatingModesGroupBox.Size = New System.Drawing.Size(616, 216)
507         Me.m_OperatingModesGroupBox.TabIndex = 0
508         Me.m_OperatingModesGroupBox.TabStop = False
509         Me.m_OperatingModesGroupBox.Text = "Operating Modes"
510         '
511         'm_SkipEngineTransmissionIfAllModesCheckBox
512         '
513         Me.m_SkipEngineTransmissionIfAllModesCheckBox.CheckAlign = System.Drawing.
        ContentAlignment.MiddleRight
514         Me.m_SkipEngineTransmissionIfAllModesCheckBox.Location = New System.Drawing.Point
        (488, 124)
515         Me.m_SkipEngineTransmissionIfAllModesCheckBox.Name =
        "m_SkipEngineTransmissionIfAllModesCheckBox"
516         Me.m_SkipEngineTransmissionIfAllModesCheckBox.Size = New System.Drawing.Size(40,
        16)
517         Me.m_SkipEngineTransmissionIfAllModesCheckBox.TabIndex = 18
518         Me.m_SkipEngineTransmissionIfAllModesCheckBox.Text = "All"
519         Me.m_SkipEngineTransmissionIfAllModesCheckBox.TextAlign = System.Drawing.
        ContentAlignment.MiddleCenter
520         Me.m_SkipEngineTransmissionIfAllModesCheckBox.ThreeState = True
```

```
521         '
522         'm_CreateTransmissionIfAllModesCheckBox
523         '
524         Me.m_CreateTransmissionIfAllModesCheckBox.CheckAlign = System.Drawing.
    ContentAlignment.MiddleRight
525         Me.m_CreateTransmissionIfAllModesCheckBox.Location = New System.Drawing.Point(488,
     76)
526         Me.m_CreateTransmissionIfAllModesCheckBox.Name =
    "m_CreateTransmissionIfAllModesCheckBox"
527         Me.m_CreateTransmissionIfAllModesCheckBox.Size = New System.Drawing.Size(40, 16)
528         Me.m_CreateTransmissionIfAllModesCheckBox.TabIndex = 14
529         Me.m_CreateTransmissionIfAllModesCheckBox.Text = "All"
530         Me.m_CreateTransmissionIfAllModesCheckBox.TextAlign = System.Drawing.
    ContentAlignment.MiddleCenter
531         Me.m_CreateTransmissionIfAllModesCheckBox.ThreeState = True
532         '
533         'm_CreateEngineIfAllModesCheckBox
534         '
535         Me.m_CreateEngineIfAllModesCheckBox.CheckAlign = System.Drawing.ContentAlignment.
    MiddleRight
536         Me.m_CreateEngineIfAllModesCheckBox.Location = New System.Drawing.Point(488, 28)
537         Me.m_CreateEngineIfAllModesCheckBox.Name = "m_CreateEngineIfAllModesCheckBox"
538         Me.m_CreateEngineIfAllModesCheckBox.Size = New System.Drawing.Size(40, 16)
539         Me.m_CreateEngineIfAllModesCheckBox.TabIndex = 10
540         Me.m_CreateEngineIfAllModesCheckBox.Text = "All"
541         Me.m_CreateEngineIfAllModesCheckBox.TextAlign = System.Drawing.ContentAlignment.
    MiddleCenter
542         Me.m_CreateEngineIfAllModesCheckBox.ThreeState = True
543         '
544         'm_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox
545         '
546         Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Location = New System
    .Drawing.Point(392, 144)
547         Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Name =
    "m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox"
548         Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Size = New System.
    Drawing.Size(152, 16)
549         Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.TabIndex = 21
550         Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Text = "Tech Enable
    Miss Match"
551         '
552         'm_SkipEngineTransmissionIfMultipleRCModeCheckBox
553         '
554         Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Location = New System.Drawing.
    Point(304, 144)
555         Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Name =
    "m_SkipEngineTransmissionIfMultipleRCModeCheckBox"
556         Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Size = New System.Drawing.Size
    (88, 16)
557         Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.TabIndex = 20
558         Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Text = "Multiple RC"
559         '
560         'm_SkipEngineTransmissionIfCarsInvolvedModeCheckBox
561         '
562         Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Location = New System.
    Drawing.Point(208, 144)
563         Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Name =
    "m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox"
564         Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Size = New System.Drawing.
    Size(96, 16)
565         Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.TabIndex = 19
566         Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Text = "Cars Involved"
567         '
568         'm_SkipEngineTransmissionIfLabel
569         '
570         Me.m_SkipEngineTransmissionIfLabel.BorderStyle = System.Windows.Forms.BorderStyle.
    Fixed3D
571         Me.m_SkipEngineTransmissionIfLabel.Location = New System.Drawing.Point(200, 120)
572         Me.m_SkipEngineTransmissionIfLabel.Name = "m_SkipEngineTransmissionIfLabel"
573         Me.m_SkipEngineTransmissionIfLabel.Size = New System.Drawing.Size(352, 44)
```

```
574         Me.m_SkipEngineTransmissionIfLabel.TabIndex = 18
575         Me.m_SkipEngineTransmissionIfLabel.Text = "Skip Engine/Transmission If:"
576         '
577         'm_MonteCarloIterationsNumericUpDown
578         '
579         Me.m_MonteCarloIterationsNumericUpDown.Location = New System.Drawing.Point(88,
    184)
580         Me.m_MonteCarloIterationsNumericUpDown.Maximum = New Decimal(New Integer()
    {10000000, 0, 0, 0})
581         Me.m_MonteCarloIterationsNumericUpDown.Name = "m_MonteCarloIterationsNumericUpDown
    "
582         Me.m_MonteCarloIterationsNumericUpDown.Size = New System.Drawing.Size(80, 20)
583         Me.m_MonteCarloIterationsNumericUpDown.TabIndex = 7
584         Me.m_MonteCarloIterationsNumericUpDown.TextAlign = System.Windows.Forms.
    HorizontalAlignment.Right
585         Me.m_MonteCarloIterationsNumericUpDown.Value = New Decimal(New Integer() {1, 0, 0,
     0})
586         '
587         'm_MonteCarloIterationsLabel
588         '
589         Me.m_MonteCarloIterationsLabel.Location = New System.Drawing.Point(32, 184)
590         Me.m_MonteCarloIterationsLabel.Name = "m_MonteCarloIterationsLabel"
591         Me.m_MonteCarloIterationsLabel.Size = New System.Drawing.Size(64, 16)
592         Me.m_MonteCarloIterationsLabel.TabIndex = 7
593         Me.m_MonteCarloIterationsLabel.Text = "Iterations:"
594         Me.m_MonteCarloIterationsLabel.TextAlign = System.Drawing.ContentAlignment.
    BottomLeft
595         '
596         'm_MonteCarloAnalysisCheckBox
597         '
598         Me.m_MonteCarloAnalysisCheckBox.Location = New System.Drawing.Point(16, 168)
599         Me.m_MonteCarloAnalysisCheckBox.Name = "m_MonteCarloAnalysisCheckBox"
600         Me.m_MonteCarloAnalysisCheckBox.Size = New System.Drawing.Size(144, 16)
601         Me.m_MonteCarloAnalysisCheckBox.TabIndex = 6
602         Me.m_MonteCarloAnalysisCheckBox.Text = "Monte Carlo Analysys"
603         '
604         'm_ReserveTechsIfInComplianceModeCheckBox
605         '
606         Me.m_ReserveTechsIfInComplianceModeCheckBox.Location = New System.Drawing.Point
    (208, 176)
607         Me.m_ReserveTechsIfInComplianceModeCheckBox.Name =
    "m_ReserveTechsIfInComplianceModeCheckBox"
608         Me.m_ReserveTechsIfInComplianceModeCheckBox.Size = New System.Drawing.Size(224,
    16)
609         Me.m_ReserveTechsIfInComplianceModeCheckBox.TabIndex = 22
610         Me.m_ReserveTechsIfInComplianceModeCheckBox.Text = "Reserve Technologies If In
    Compliance"
611         '
612         'm_StageAnalysisCheckBox
613         '
614         Me.m_StageAnalysisCheckBox.Location = New System.Drawing.Point(16, 144)
615         Me.m_StageAnalysisCheckBox.Name = "m_StageAnalysisCheckBox"
616         Me.m_StageAnalysisCheckBox.Size = New System.Drawing.Size(104, 16)
617         Me.m_StageAnalysisCheckBox.TabIndex = 5
618         Me.m_StageAnalysisCheckBox.Text = "Stage Analysis"
619         '
620         'm_AutoInvokeEfficiencyModeCheckBox
621         '
622         Me.m_AutoInvokeEfficiencyModeCheckBox.Location = New System.Drawing.Point(16, 96)
623         Me.m_AutoInvokeEfficiencyModeCheckBox.Name = "m_AutoInvokeEfficiencyModeCheckBox"
624         Me.m_AutoInvokeEfficiencyModeCheckBox.Size = New System.Drawing.Size(176, 16)
625         Me.m_AutoInvokeEfficiencyModeCheckBox.TabIndex = 3
626         Me.m_AutoInvokeEfficiencyModeCheckBox.Text = "Auto-invoke Efficiency Model"
627         '
628         'm_SelectAllModesCheckBox
629         '
630         Me.m_SelectAllModesCheckBox.Location = New System.Drawing.Point(16, 24)
631         Me.m_SelectAllModesCheckBox.Name = "m_SelectAllModesCheckBox"
632         Me.m_SelectAllModesCheckBox.Size = New System.Drawing.Size(120, 16)
633         Me.m_SelectAllModesCheckBox.TabIndex = 0
```

```
634         Me.m_SelectAllModesCheckBox.Text = "Select &All Modes"
635         Me.m_SelectAllModesCheckBox.ThreeState = True
636         '
637         'm_CreateTransmissionIfTechEnableMissMatchModeCheckBox
638         '
639         Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Location = New System.
    Drawing.Point(392, 96)
640         Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Name =
    "m_CreateTransmissionIfTechEnableMissMatchModeCheckBox"
641         Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Size = New System.Drawing
    .Size(152, 16)
642         Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.TabIndex = 17
643         Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Text = "Tech Enable Miss
    Match"
644         '
645         'm_CreateTransmissionIfMultipleRCModeCheckBox
646         '
647         Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Location = New System.Drawing.
    Point(304, 96)
648         Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Name =
    "m_CreateTransmissionIfMultipleRCModeCheckBox"
649         Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Size = New System.Drawing.Size(88,
     16)
650         Me.m_CreateTransmissionIfMultipleRCModeCheckBox.TabIndex = 16
651         Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Text = "Multiple RC"
652         '
653         'm_CreateTransmissionIfCarsInvolvedModeCheckBox
654         '
655         Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Location = New System.Drawing.
    Point(208, 96)
656         Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Name =
    "m_CreateTransmissionIfCarsInvolvedModeCheckBox"
657         Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Size = New System.Drawing.Size
    (96, 16)
658         Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.TabIndex = 15
659         Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Text = "Cars Involved"
660         '
661         'm_CreateEngineIfTechEnableMissMatchModeCheckBox
662         '
663         Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Location = New System.Drawing.
    Point(392, 48)
664         Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Name =
    "m_CreateEngineIfTechEnableMissMatchModeCheckBox"
665         Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Size = New System.Drawing.Size
    (152, 16)
666         Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.TabIndex = 13
667         Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Text = "Tech Enable Miss Match"
668         '
669         'm_CreateEngineIfMultipleRCModeCheckBox
670         '
671         Me.m_CreateEngineIfMultipleRCModeCheckBox.Location = New System.Drawing.Point(304,
     48)
672         Me.m_CreateEngineIfMultipleRCModeCheckBox.Name =
    "m_CreateEngineIfMultipleRCModeCheckBox"
673         Me.m_CreateEngineIfMultipleRCModeCheckBox.Size = New System.Drawing.Size(88, 16)
674         Me.m_CreateEngineIfMultipleRCModeCheckBox.TabIndex = 12
675         Me.m_CreateEngineIfMultipleRCModeCheckBox.Text = "Multiple RC"
676         '
677         'm_CreateEngineIfCarsInvolvedModeCheckBox
678         '
679         Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Location = New System.Drawing.Point
    (208, 48)
680         Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Name =
    "m_CreateEngineIfCarsInvolvedModeCheckBox"
681         Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Size = New System.Drawing.Size(96, 16)
682         Me.m_CreateEngineIfCarsInvolvedModeCheckBox.TabIndex = 11
683         Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Text = "Cars Involved"
684         '
685         'm_BackFillTechnologyModeCheckBox
686         '
```

```
687         Me.m_BackFillTechnologyModeCheckBox.Location = New System.Drawing.Point(16, 120)
688         Me.m_BackFillTechnologyModeCheckBox.Name = "m_BackFillTechnologyModeCheckBox"
689         Me.m_BackFillTechnologyModeCheckBox.Size = New System.Drawing.Size(128, 16)
690         Me.m_BackFillTechnologyModeCheckBox.TabIndex = 4
691         Me.m_BackFillTechnologyModeCheckBox.Text = "Back Fill Technology"
692         '
693         'm_LowCostTechnologyFirstModeCheckBox
694         '
695         Me.m_LowCostTechnologyFirstModeCheckBox.Location = New System.Drawing.Point(16,
    72)
696         Me.m_LowCostTechnologyFirstModeCheckBox.Name =
    "m_LowCostTechnologyFirstModeCheckBox"
697         Me.m_LowCostTechnologyFirstModeCheckBox.Size = New System.Drawing.Size(160, 16)
698         Me.m_LowCostTechnologyFirstModeCheckBox.TabIndex = 2
699         Me.m_LowCostTechnologyFirstModeCheckBox.Text = "Low Cost Technology First"
700         '
701         'm_NoFinesModeCheckBox
702         '
703         Me.m_NoFinesModeCheckBox.Location = New System.Drawing.Point(16, 48)
704         Me.m_NoFinesModeCheckBox.Name = "m_NoFinesModeCheckBox"
705         Me.m_NoFinesModeCheckBox.Size = New System.Drawing.Size(72, 16)
706         Me.m_NoFinesModeCheckBox.TabIndex = 1
707         Me.m_NoFinesModeCheckBox.Text = "No Fines"
708         '
709         'm_CreateEngineIfLabel
710         '
711         Me.m_CreateEngineIfLabel.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D
712         Me.m_CreateEngineIfLabel.Location = New System.Drawing.Point(200, 24)
713         Me.m_CreateEngineIfLabel.Name = "m_CreateEngineIfLabel"
714         Me.m_CreateEngineIfLabel.Size = New System.Drawing.Size(352, 44)
715         Me.m_CreateEngineIfLabel.TabIndex = 10
716         Me.m_CreateEngineIfLabel.Text = "Create Engine If:"
717         '
718         'm_CreateTransmissionIfLabel
719         '
720         Me.m_CreateTransmissionIfLabel.BorderStyle = System.Windows.Forms.BorderStyle.
    Fixed3D
721         Me.m_CreateTransmissionIfLabel.Location = New System.Drawing.Point(200, 72)
722         Me.m_CreateTransmissionIfLabel.Name = "m_CreateTransmissionIfLabel"
723         Me.m_CreateTransmissionIfLabel.Size = New System.Drawing.Size(352, 44)
724         Me.m_CreateTransmissionIfLabel.TabIndex = 15
725         Me.m_CreateTransmissionIfLabel.Text = "Create Transmission If:"
726         '
727         'm_EncryptionSettingsTabPage
728         '
729         Me.m_EncryptionSettingsTabPage.Controls.Add(Me.m_EncryptionSettingsSavingGroupBox)
730         Me.m_EncryptionSettingsTabPage.Controls.Add(Me.
    m_EncryptionSettingsLoadingGroupBox)
731         Me.m_EncryptionSettingsTabPage.Location = New System.Drawing.Point(4, 22)
732         Me.m_EncryptionSettingsTabPage.Name = "m_EncryptionSettingsTabPage"
733         Me.m_EncryptionSettingsTabPage.Size = New System.Drawing.Size(630, 238)
734         Me.m_EncryptionSettingsTabPage.TabIndex = 3
735         Me.m_EncryptionSettingsTabPage.Text = "Encryption Settings"
736         '
737         'm_EncryptionSettingsSavingGroupBox
738         '
739         Me.m_EncryptionSettingsSavingGroupBox.Controls.Add(Me.m_EncryptOutputFilesPanel)
740         Me.m_EncryptionSettingsSavingGroupBox.Controls.Add(Me.
    m_EncryptOutputFilesCheckBox)
741         Me.m_EncryptionSettingsSavingGroupBox.Location = New System.Drawing.Point(8, 8)
742         Me.m_EncryptionSettingsSavingGroupBox.Name = "m_EncryptionSettingsSavingGroupBox"
743         Me.m_EncryptionSettingsSavingGroupBox.Size = New System.Drawing.Size(304, 216)
744         Me.m_EncryptionSettingsSavingGroupBox.TabIndex = 0
745         Me.m_EncryptionSettingsSavingGroupBox.TabStop = False
746         Me.m_EncryptionSettingsSavingGroupBox.Text = "File Saving Settings"
747         '
748         'm_EncryptOutputFilesPanel
749         '
750         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_EncryptDocPropertiesCheckBox)
751         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_PasswordToModifyTextBox)
```

```
752         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_PasswordToModifyLabel)
753         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_KeyLengthLabel)
754         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_EncryptionProviderLabel)
755         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_PasswordToOpenTextBox)
756         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_PasswordToModifyCheckBox)
757         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_EncryptionProviderComboBox)
758         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_KeyLengthNumericUpDown)
759         Me.m_EncryptOutputFilesPanel.Controls.Add(Me.m_PasswordToOpenLabel)
760         Me.m_EncryptOutputFilesPanel.Location = New System.Drawing.Point(8, 40)
761         Me.m_EncryptOutputFilesPanel.Name = "m_EncryptOutputFilesPanel"
762         Me.m_EncryptOutputFilesPanel.Size = New System.Drawing.Size(288, 136)
763         Me.m_EncryptOutputFilesPanel.TabIndex = 1
764         '
765         'm_EncryptDocPropertiesCheckBox
766         '
767         Me.m_EncryptDocPropertiesCheckBox.Location = New System.Drawing.Point(8, 48)
768         Me.m_EncryptDocPropertiesCheckBox.Name = "m_EncryptDocPropertiesCheckBox"
769         Me.m_EncryptDocPropertiesCheckBox.Size = New System.Drawing.Size(176, 16)
770         Me.m_EncryptDocPropertiesCheckBox.TabIndex = 4
771         Me.m_EncryptDocPropertiesCheckBox.Text = "Encrypt Document Properties"
772         '
773         'm_PasswordToModifyTextBox
774         '
775         Me.m_PasswordToModifyTextBox.Location = New System.Drawing.Point(152, 88)
776         Me.m_PasswordToModifyTextBox.MaxLength = 32
777         Me.m_PasswordToModifyTextBox.Name = "m_PasswordToModifyTextBox"
778         Me.m_PasswordToModifyTextBox.PasswordChar = Microsoft.VisualBasic.ChrW(42)
779         Me.m_PasswordToModifyTextBox.Size = New System.Drawing.Size(128, 20)
780         Me.m_PasswordToModifyTextBox.TabIndex = 8
781         Me.m_PasswordToModifyTextBox.Text = ""
782         '
783         'm_PasswordToModifyLabel
784         '
785         Me.m_PasswordToModifyLabel.Location = New System.Drawing.Point(152, 72)
786         Me.m_PasswordToModifyLabel.Name = "m_PasswordToModifyLabel"
787         Me.m_PasswordToModifyLabel.Size = New System.Drawing.Size(112, 16)
788         Me.m_PasswordToModifyLabel.TabIndex = 7
789         Me.m_PasswordToModifyLabel.Text = "Password To Modify:"
790         Me.m_PasswordToModifyLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
791         '
792         'm_KeyLengthLabel
793         '
794         Me.m_KeyLengthLabel.Location = New System.Drawing.Point(216, 8)
795         Me.m_KeyLengthLabel.Name = "m_KeyLengthLabel"
796         Me.m_KeyLengthLabel.Size = New System.Drawing.Size(64, 16)
797         Me.m_KeyLengthLabel.TabIndex = 2
798         Me.m_KeyLengthLabel.Text = "Key Length"
799         '
800         'm_EncryptionProviderLabel
801         '
802         Me.m_EncryptionProviderLabel.Location = New System.Drawing.Point(8, 8)
803         Me.m_EncryptionProviderLabel.Name = "m_EncryptionProviderLabel"
804         Me.m_EncryptionProviderLabel.Size = New System.Drawing.Size(200, 16)
805         Me.m_EncryptionProviderLabel.TabIndex = 0
806         Me.m_EncryptionProviderLabel.Text = "Select Encryption Provider"
807         '
808         'm_PasswordToOpenTextBox
809         '
810         Me.m_PasswordToOpenTextBox.Location = New System.Drawing.Point(8, 88)
811         Me.m_PasswordToOpenTextBox.MaxLength = 32
812         Me.m_PasswordToOpenTextBox.Name = "m_PasswordToOpenTextBox"
813         Me.m_PasswordToOpenTextBox.PasswordChar = Microsoft.VisualBasic.ChrW(42)
814         Me.m_PasswordToOpenTextBox.Size = New System.Drawing.Size(136, 20)
815         Me.m_PasswordToOpenTextBox.TabIndex = 6
816         Me.m_PasswordToOpenTextBox.Text = ""
817         '
818         'm_PasswordToModifyCheckBox
819         '
820         Me.m_PasswordToModifyCheckBox.Font = New System.Drawing.Font("Microsoft Sans Serif ↙
    ", 8.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0 ↙
```

```
, Byte))
821         Me.m_PasswordToModifyCheckBox.Location = New System.Drawing.Point(152, 112)
822         Me.m_PasswordToModifyCheckBox.Name = "m_PasswordToModifyCheckBox"
823         Me.m_PasswordToModifyCheckBox.Size = New System.Drawing.Size(120, 16)
824         Me.m_PasswordToModifyCheckBox.TabIndex = 9
825         Me.m_PasswordToModifyCheckBox.Text = "Same as Open PW"
826         '
827         'm_EncryptionProviderComboBox
828         '
829         Me.m_EncryptionProviderComboBox.DropDownStyle = System.Windows.Forms.ComboBoxStyle ↙
    .DropDownList
830         Me.m_EncryptionProviderComboBox.DropDownWidth = 350
831         Me.m_EncryptionProviderComboBox.Location = New System.Drawing.Point(8, 24)
832         Me.m_EncryptionProviderComboBox.Name = "m_EncryptionProviderComboBox"
833         Me.m_EncryptionProviderComboBox.Size = New System.Drawing.Size(208, 21)
834         Me.m_EncryptionProviderComboBox.TabIndex = 1
835         '
836         'm_KeyLengthNumericUpDown
837         '
838         Me.m_KeyLengthNumericUpDown.Increment = New Decimal(New Integer() {8, 0, 0, 0})
839         Me.m_KeyLengthNumericUpDown.Location = New System.Drawing.Point(216, 24)
840         Me.m_KeyLengthNumericUpDown.Maximum = New Decimal(New Integer() {128, 0, 0, 0})
841         Me.m_KeyLengthNumericUpDown.Minimum = New Decimal(New Integer() {64, 0, 0, 0})
842         Me.m_KeyLengthNumericUpDown.Name = "m_KeyLengthNumericUpDown"
843         Me.m_KeyLengthNumericUpDown.RightToLeft = System.Windows.Forms.RightToLeft.No
844         Me.m_KeyLengthNumericUpDown.Size = New System.Drawing.Size(64, 20)
845         Me.m_KeyLengthNumericUpDown.TabIndex = 3
846         Me.m_KeyLengthNumericUpDown.TextAlign = System.Windows.Forms.HorizontalAlignment. ↙
    Right
847         Me.m_KeyLengthNumericUpDown.Value = New Decimal(New Integer() {128, 0, 0, 0})
848         '
849         'm_PasswordToOpenLabel
850         '
851         Me.m_PasswordToOpenLabel.Location = New System.Drawing.Point(8, 72)
852         Me.m_PasswordToOpenLabel.Name = "m_PasswordToOpenLabel"
853         Me.m_PasswordToOpenLabel.Size = New System.Drawing.Size(112, 16)
854         Me.m_PasswordToOpenLabel.TabIndex = 5
855         Me.m_PasswordToOpenLabel.Text = "Password To Open:"
856         Me.m_PasswordToOpenLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
857         '
858         'm_EncryptOutputFilesCheckBox
859         '
860         Me.m_EncryptOutputFilesCheckBox.Location = New System.Drawing.Point(16, 24)
861         Me.m_EncryptOutputFilesCheckBox.Name = "m_EncryptOutputFilesCheckBox"
862         Me.m_EncryptOutputFilesCheckBox.Size = New System.Drawing.Size(192, 16)
863         Me.m_EncryptOutputFilesCheckBox.TabIndex = 0
864         Me.m_EncryptOutputFilesCheckBox.Text = "Encrypt Output Files and Reports"
865         '
866         'm_EncryptionSettingsLoadingGroupBox
867         '
868         Me.m_EncryptionSettingsLoadingGroupBox.Controls.Add(Me.m_ReadEncryptedFilesPanel)
869         Me.m_EncryptionSettingsLoadingGroupBox.Controls.Add(Me. ↙
    m_ReadEncryptedFilesCheckBox)
870         Me.m_EncryptionSettingsLoadingGroupBox.Location = New System.Drawing.Point(320, 8)
871         Me.m_EncryptionSettingsLoadingGroupBox.Name = "m_EncryptionSettingsLoadingGroupBox ↙
    "
872         Me.m_EncryptionSettingsLoadingGroupBox.Size = New System.Drawing.Size(304, 216)
873         Me.m_EncryptionSettingsLoadingGroupBox.TabIndex = 1
874         Me.m_EncryptionSettingsLoadingGroupBox.TabStop = False
875         Me.m_EncryptionSettingsLoadingGroupBox.Text = "File Loading Settings"
876         '
877         'm_ReadEncryptedFilesPanel
878         '
879         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputMultiPwTechTextBox)
880         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputMultiPwTechLabel)
881         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputMultiPwTranTextBox)
882         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputMultiPwTranLabel)
883         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputMultiPwEngTextBox)
884         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputMultiPwEngLabel)
885         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputMultiPwVehTextBox)
```

```
886         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputMultiPwVehLabel)
887         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputSinglePwTextBox)
888         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputMultiPwCheckBox)
889         Me.m_ReadEncryptedFilesPanel.Controls.Add(Me.m_InputSinglePwCheckBox)
890         Me.m_ReadEncryptedFilesPanel.Location = New System.Drawing.Point(8, 40)
891         Me.m_ReadEncryptedFilesPanel.Name = "m_ReadEncryptedFilesPanel"
892         Me.m_ReadEncryptedFilesPanel.Size = New System.Drawing.Size(288, 136)
893         Me.m_ReadEncryptedFilesPanel.TabIndex = 1
894         '
895         'm_InputMultiPwTechTextBox
896         '
897         Me.m_InputMultiPwTechTextBox.Location = New System.Drawing.Point(152, 104)
898         Me.m_InputMultiPwTechTextBox.MaxLength = 32
899         Me.m_InputMultiPwTechTextBox.Name = "m_InputMultiPwTechTextBox"
900         Me.m_InputMultiPwTechTextBox.PasswordChar = Microsoft.VisualBasic.ChrW(42)
901         Me.m_InputMultiPwTechTextBox.Size = New System.Drawing.Size(128, 20)
902         Me.m_InputMultiPwTechTextBox.TabIndex = 11
903         Me.m_InputMultiPwTechTextBox.Text = ""
904         '
905         'm_InputMultiPwTechLabel
906         '
907         Me.m_InputMultiPwTechLabel.Location = New System.Drawing.Point(152, 88)
908         Me.m_InputMultiPwTechLabel.Name = "m_InputMultiPwTechLabel"
909         Me.m_InputMultiPwTechLabel.Size = New System.Drawing.Size(112, 16)
910         Me.m_InputMultiPwTechLabel.TabIndex = 10
911         Me.m_InputMultiPwTechLabel.Text = "Technologies PW"
912         '
913         'm_InputMultiPwTranTextBox
914         '
915         Me.m_InputMultiPwTranTextBox.Location = New System.Drawing.Point(8, 104)
916         Me.m_InputMultiPwTranTextBox.MaxLength = 32
917         Me.m_InputMultiPwTranTextBox.Name = "m_InputMultiPwTranTextBox"
918         Me.m_InputMultiPwTranTextBox.PasswordChar = Microsoft.VisualBasic.ChrW(42)
919         Me.m_InputMultiPwTranTextBox.Size = New System.Drawing.Size(136, 20)
920         Me.m_InputMultiPwTranTextBox.TabIndex = 9
921         Me.m_InputMultiPwTranTextBox.Text = ""
922         '
923         'm_InputMultiPwTranLabel
924         '
925         Me.m_InputMultiPwTranLabel.Location = New System.Drawing.Point(8, 88)
926         Me.m_InputMultiPwTranLabel.Name = "m_InputMultiPwTranLabel"
927         Me.m_InputMultiPwTranLabel.Size = New System.Drawing.Size(112, 16)
928         Me.m_InputMultiPwTranLabel.TabIndex = 8
929         Me.m_InputMultiPwTranLabel.Text = "Transmissions PW"
930         '
931         'm_InputMultiPwEngTextBox
932         '
933         Me.m_InputMultiPwEngTextBox.Location = New System.Drawing.Point(152, 64)
934         Me.m_InputMultiPwEngTextBox.MaxLength = 32
935         Me.m_InputMultiPwEngTextBox.Name = "m_InputMultiPwEngTextBox"
936         Me.m_InputMultiPwEngTextBox.PasswordChar = Microsoft.VisualBasic.ChrW(42)
937         Me.m_InputMultiPwEngTextBox.Size = New System.Drawing.Size(128, 20)
938         Me.m_InputMultiPwEngTextBox.TabIndex = 7
939         Me.m_InputMultiPwEngTextBox.Text = ""
940         '
941         'm_InputMultiPwEngLabel
942         '
943         Me.m_InputMultiPwEngLabel.Location = New System.Drawing.Point(152, 48)
944         Me.m_InputMultiPwEngLabel.Name = "m_InputMultiPwEngLabel"
945         Me.m_InputMultiPwEngLabel.Size = New System.Drawing.Size(104, 16)
946         Me.m_InputMultiPwEngLabel.TabIndex = 6
947         Me.m_InputMultiPwEngLabel.Text = "Engines File PW"
948         '
949         'm_InputMultiPwVehTextBox
950         '
951         Me.m_InputMultiPwVehTextBox.Location = New System.Drawing.Point(8, 64)
952         Me.m_InputMultiPwVehTextBox.MaxLength = 32
953         Me.m_InputMultiPwVehTextBox.Name = "m_InputMultiPwVehTextBox"
954         Me.m_InputMultiPwVehTextBox.PasswordChar = Microsoft.VisualBasic.ChrW(42)
955         Me.m_InputMultiPwVehTextBox.Size = New System.Drawing.Size(136, 20)
```

```
956         Me.m_InputMultiPwVehTextBox.TabIndex = 5
957         Me.m_InputMultiPwVehTextBox.Text = ""
958         '
959         'm_InputMultiPwVehLabel
960         '
961         Me.m_InputMultiPwVehLabel.Location = New System.Drawing.Point(8, 48)
962         Me.m_InputMultiPwVehLabel.Name = "m_InputMultiPwVehLabel"
963         Me.m_InputMultiPwVehLabel.Size = New System.Drawing.Size(112, 16)
964         Me.m_InputMultiPwVehLabel.TabIndex = 4
965         Me.m_InputMultiPwVehLabel.Text = "Vehicles File PW"
966         '
967         'm_InputSinglePwTextBox
968         '
969         Me.m_InputSinglePwTextBox.Location = New System.Drawing.Point(152, 8)
970         Me.m_InputSinglePwTextBox.MaxLength = 32
971         Me.m_InputSinglePwTextBox.Name = "m_InputSinglePwTextBox"
972         Me.m_InputSinglePwTextBox.PasswordChar = Microsoft.VisualBasic.ChrW(42)
973         Me.m_InputSinglePwTextBox.Size = New System.Drawing.Size(128, 20)
974         Me.m_InputSinglePwTextBox.TabIndex = 1
975         Me.m_InputSinglePwTextBox.Text = ""
976         '
977         'm_InputMultiPwCheckBox
978         '
979         Me.m_InputMultiPwCheckBox.Checked = True
980         Me.m_InputMultiPwCheckBox.CheckState = System.Windows.Forms.CheckState.Checked
981         Me.m_InputMultiPwCheckBox.Location = New System.Drawing.Point(8, 29)
982         Me.m_InputMultiPwCheckBox.Name = "m_InputMultiPwCheckBox"
983         Me.m_InputMultiPwCheckBox.Size = New System.Drawing.Size(160, 16)
984         Me.m_InputMultiPwCheckBox.TabIndex = 2
985         Me.m_InputMultiPwCheckBox.Text = "Use Individual Passwords"
986         '
987         'm_InputSinglePwCheckBox
988         '
989         Me.m_InputSinglePwCheckBox.Checked = True
990         Me.m_InputSinglePwCheckBox.CheckState = System.Windows.Forms.CheckState.Checked
991         Me.m_InputSinglePwCheckBox.Location = New System.Drawing.Point(8, 10)
992         Me.m_InputSinglePwCheckBox.Name = "m_InputSinglePwCheckBox"
993         Me.m_InputSinglePwCheckBox.Size = New System.Drawing.Size(152, 16)
994         Me.m_InputSinglePwCheckBox.TabIndex = 0
995         Me.m_InputSinglePwCheckBox.Text = "Use One PW For All Files"
996         '
997         'm_ReadEncryptedFilesCheckBox
998         '
999         Me.m_ReadEncryptedFilesCheckBox.Location = New System.Drawing.Point(16, 24)
1000        Me.m_ReadEncryptedFilesCheckBox.Name = "m_ReadEncryptedFilesCheckBox"
1001        Me.m_ReadEncryptedFilesCheckBox.Size = New System.Drawing.Size(160, 16)
1002        Me.m_ReadEncryptedFilesCheckBox.TabIndex = 0
1003        Me.m_ReadEncryptedFilesCheckBox.Text = "Read Encypted Input Files"
1004        '
1005        'm_OutputSettingsTabPage
1006        '
1007        Me.m_OutputSettingsTabPage.Controls.Add(Me.m_TemplateSettingsGroupBox)
1008        Me.m_OutputSettingsTabPage.Controls.Add(Me.m_ReportSettingsGroupBox)
1009        Me.m_OutputSettingsTabPage.Controls.Add(Me.m_GeneralGroupBox)
1010        Me.m_OutputSettingsTabPage.Location = New System.Drawing.Point(4, 22)
1011        Me.m_OutputSettingsTabPage.Name = "m_OutputSettingsTabPage"
1012        Me.m_OutputSettingsTabPage.Size = New System.Drawing.Size(630, 238)
1013        Me.m_OutputSettingsTabPage.TabIndex = 4
1014        Me.m_OutputSettingsTabPage.Text = "Output Settings"
1015        '
1016        'm_TemplateSettingsGroupBox
1017        '
1018        Me.m_TemplateSettingsGroupBox.Controls.Add(Me.m_DefaultOutputTemplateBrowseButton)
1019        Me.m_TemplateSettingsGroupBox.Controls.Add(Me.m_DefaultOutputTemplateTextBox)
1020        Me.m_TemplateSettingsGroupBox.Controls.Add(Me.m_ConfigureTemplatesButton)
1021        Me.m_TemplateSettingsGroupBox.Controls.Add(Me.m_UseOutputTemplatesCheckBox)
1022        Me.m_TemplateSettingsGroupBox.Location = New System.Drawing.Point(8, 120)
1023        Me.m_TemplateSettingsGroupBox.Name = "m_TemplateSettingsGroupBox"
1024        Me.m_TemplateSettingsGroupBox.Size = New System.Drawing.Size(304, 104)
1025        Me.m_TemplateSettingsGroupBox.TabIndex = 2
```

```
1026        Me.m_TemplateSettingsGroupBox.TabStop = False
1027        Me.m_TemplateSettingsGroupBox.Text = "Template Settings"
1028        '
1029        'm_DefaultOutputTemplateBrowseButton
1030        '
1031        Me.m_DefaultOutputTemplateBrowseButton.Location = New System.Drawing.Point(208,
    72)
1032        Me.m_DefaultOutputTemplateBrowseButton.Name = "m_DefaultOutputTemplateBrowseButton
    "
1033        Me.m_DefaultOutputTemplateBrowseButton.Size = New System.Drawing.Size(80, 20)
1034        Me.m_DefaultOutputTemplateBrowseButton.TabIndex = 4
1035        Me.m_DefaultOutputTemplateBrowseButton.Text = "Browse..."
1036        '
1037        'm_DefaultOutputTemplateTextBox
1038        '
1039        Me.m_DefaultOutputTemplateTextBox.BackColor = System.Drawing.SystemColors.Window
1040        Me.m_DefaultOutputTemplateTextBox.Location = New System.Drawing.Point(16, 48)
1041        Me.m_DefaultOutputTemplateTextBox.Name = "m_DefaultOutputTemplateTextBox"
1042        Me.m_DefaultOutputTemplateTextBox.ReadOnly = True
1043        Me.m_DefaultOutputTemplateTextBox.Size = New System.Drawing.Size(272, 20)
1044        Me.m_DefaultOutputTemplateTextBox.TabIndex = 3
1045        Me.m_DefaultOutputTemplateTextBox.Text = ""
1046        '
1047        'm_ConfigureTemplatesButton
1048        '
1049        Me.m_ConfigureTemplatesButton.Location = New System.Drawing.Point(208, 24)
1050        Me.m_ConfigureTemplatesButton.Name = "m_ConfigureTemplatesButton"
1051        Me.m_ConfigureTemplatesButton.Size = New System.Drawing.Size(80, 20)
1052        Me.m_ConfigureTemplatesButton.TabIndex = 1
1053        Me.m_ConfigureTemplatesButton.Text = "Confi&gure"
1054        Me.m_ConfigureTemplatesButton.Visible = False
1055        '
1056        'm_UseOutputTemplatesCheckBox
1057        '
1058        Me.m_UseOutputTemplatesCheckBox.Location = New System.Drawing.Point(16, 24)
1059        Me.m_UseOutputTemplatesCheckBox.Name = "m_UseOutputTemplatesCheckBox"
1060        Me.m_UseOutputTemplatesCheckBox.Size = New System.Drawing.Size(192, 16)
1061        Me.m_UseOutputTemplatesCheckBox.TabIndex = 0
1062        Me.m_UseOutputTemplatesCheckBox.Text = "Use Templates for Output"
1063        '
1064        'm_ReportSettingsGroupBox
1065        '
1066        Me.m_ReportSettingsGroupBox.Controls.Add(Me.m_GenEAReportCheckBox)
1067        Me.m_ReportSettingsGroupBox.Controls.Add(Me.m_GenAllReportsCheckBox)
1068        Me.m_ReportSettingsGroupBox.Controls.Add(Me.m_GenVehiclesReportCheckBox)
1069        Me.m_ReportSettingsGroupBox.Controls.Add(Me.m_GenMfgsReportCheckBox)
1070        Me.m_ReportSettingsGroupBox.Controls.Add(Me.m_GenIndustryReportCheckBox)
1071        Me.m_ReportSettingsGroupBox.Controls.Add(Me.m_GenEffectsReportCheckBox)
1072        Me.m_ReportSettingsGroupBox.Location = New System.Drawing.Point(320, 8)
1073        Me.m_ReportSettingsGroupBox.Name = "m_ReportSettingsGroupBox"
1074        Me.m_ReportSettingsGroupBox.Size = New System.Drawing.Size(304, 216)
1075        Me.m_ReportSettingsGroupBox.TabIndex = 3
1076        Me.m_ReportSettingsGroupBox.TabStop = False
1077        Me.m_ReportSettingsGroupBox.Text = "Report Settings"
1078        '
1079        'm_GenEAReportCheckBox
1080        '
1081        Me.m_GenEAReportCheckBox.Location = New System.Drawing.Point(16, 144)
1082        Me.m_GenEAReportCheckBox.Name = "m_GenEAReportCheckBox"
1083        Me.m_GenEAReportCheckBox.Size = New System.Drawing.Size(248, 16)
1084        Me.m_GenEAReportCheckBox.TabIndex = 5
1085        Me.m_GenEAReportCheckBox.Text = "Generate Environmental Assessment Report"
1086        '
1087        'm_GenAllReportsCheckBox
1088        '
1089        Me.m_GenAllReportsCheckBox.Location = New System.Drawing.Point(16, 24)
1090        Me.m_GenAllReportsCheckBox.Name = "m_GenAllReportsCheckBox"
1091        Me.m_GenAllReportsCheckBox.Size = New System.Drawing.Size(184, 16)
1092        Me.m_GenAllReportsCheckBox.TabIndex = 0
1093        Me.m_GenAllReportsCheckBox.Text = "Generate All Reports"
```

```
1094        Me.m_GenAllReportsCheckBox.ThreeState = True
1095        '
1096        'm_GenVehiclesReportCheckBox
1097        '
1098        Me.m_GenVehiclesReportCheckBox.Location = New System.Drawing.Point(16, 96)
1099        Me.m_GenVehiclesReportCheckBox.Name = "m_GenVehiclesReportCheckBox"
1100        Me.m_GenVehiclesReportCheckBox.Size = New System.Drawing.Size(184, 16)
1101        Me.m_GenVehiclesReportCheckBox.TabIndex = 3
1102        Me.m_GenVehiclesReportCheckBox.Text = "Generate Vehicles Report"
1103        '
1104        'm_GenMfgsReportCheckBox
1105        '
1106        Me.m_GenMfgsReportCheckBox.Location = New System.Drawing.Point(16, 72)
1107        Me.m_GenMfgsReportCheckBox.Name = "m_GenMfgsReportCheckBox"
1108        Me.m_GenMfgsReportCheckBox.Size = New System.Drawing.Size(184, 16)
1109        Me.m_GenMfgsReportCheckBox.TabIndex = 2
1110        Me.m_GenMfgsReportCheckBox.Text = "Generate Manufacturers Report"
1111        '
1112        'm_GenIndustryReportCheckBox
1113        '
1114        Me.m_GenIndustryReportCheckBox.Location = New System.Drawing.Point(16, 48)
1115        Me.m_GenIndustryReportCheckBox.Name = "m_GenIndustryReportCheckBox"
1116        Me.m_GenIndustryReportCheckBox.Size = New System.Drawing.Size(184, 16)
1117        Me.m_GenIndustryReportCheckBox.TabIndex = 1
1118        Me.m_GenIndustryReportCheckBox.Text = "Generate Industry Report"
1119        '
1120        'm_GenEffectsReportCheckBox
1121        '
1122        Me.m_GenEffectsReportCheckBox.Location = New System.Drawing.Point(16, 120)
1123        Me.m_GenEffectsReportCheckBox.Name = "m_GenEffectsReportCheckBox"
1124        Me.m_GenEffectsReportCheckBox.Size = New System.Drawing.Size(184, 16)
1125        Me.m_GenEffectsReportCheckBox.TabIndex = 4
1126        Me.m_GenEffectsReportCheckBox.Text = "Generate Effects Report"
1127        '
1128        'm_GeneralGroupBox
1129        '
1130        Me.m_GeneralGroupBox.Controls.Add(Me.m_DefaultOutputPathTextBox)
1131        Me.m_GeneralGroupBox.Controls.Add(Me.m_DefaultOutputPathLabel)
1132        Me.m_GeneralGroupBox.Controls.Add(Me.m_DefaultOutputPathBrowseButton)
1133        Me.m_GeneralGroupBox.Controls.Add(Me.m_HideUnusedRCsCheckBox)
1134        Me.m_GeneralGroupBox.Location = New System.Drawing.Point(8, 8)
1135        Me.m_GeneralGroupBox.Name = "m_GeneralGroupBox"
1136        Me.m_GeneralGroupBox.Size = New System.Drawing.Size(304, 104)
1137        Me.m_GeneralGroupBox.TabIndex = 1
1138        Me.m_GeneralGroupBox.TabStop = False
1139        Me.m_GeneralGroupBox.Text = "General"
1140        '
1141        'm_DefaultOutputPathTextBox
1142        '
1143        Me.m_DefaultOutputPathTextBox.Location = New System.Drawing.Point(16, 40)
1144        Me.m_DefaultOutputPathTextBox.Name = "m_DefaultOutputPathTextBox"
1145        Me.m_DefaultOutputPathTextBox.Size = New System.Drawing.Size(272, 20)
1146        Me.m_DefaultOutputPathTextBox.TabIndex = 1
1147        Me.m_DefaultOutputPathTextBox.Text = ""
1148        '
1149        'm_DefaultOutputPathLabel
1150        '
1151        Me.m_DefaultOutputPathLabel.Location = New System.Drawing.Point(16, 24)
1152        Me.m_DefaultOutputPathLabel.Name = "m_DefaultOutputPathLabel"
1153        Me.m_DefaultOutputPathLabel.Size = New System.Drawing.Size(232, 16)
1154        Me.m_DefaultOutputPathLabel.TabIndex = 0
1155        Me.m_DefaultOutputPathLabel.Text = "Default Output Path"
1156        '
1157        'm_DefaultOutputPathBrowseButton
1158        '
1159        Me.m_DefaultOutputPathBrowseButton.Location = New System.Drawing.Point(208, 64)
1160        Me.m_DefaultOutputPathBrowseButton.Name = "m_DefaultOutputPathBrowseButton"
1161        Me.m_DefaultOutputPathBrowseButton.Size = New System.Drawing.Size(80, 20)
1162        Me.m_DefaultOutputPathBrowseButton.TabIndex = 2
1163        Me.m_DefaultOutputPathBrowseButton.Text = "&Browse..."
```

```
1164        '
1165        'm_HideUnusedRCsCheckBox
1166        '
1167        Me.m_HideUnusedRCsCheckBox.Location = New System.Drawing.Point(16, 72)
1168        Me.m_HideUnusedRCsCheckBox.Name = "m_HideUnusedRCsCheckBox"
1169        Me.m_HideUnusedRCsCheckBox.Size = New System.Drawing.Size(120, 16)
1170        Me.m_HideUnusedRCsCheckBox.TabIndex = 3
1171        Me.m_HideUnusedRCsCheckBox.Text = "Hide Unused RCs"
1172        '
1173        'm_OtherTabPage
1174        '
1175        Me.m_OtherTabPage.Controls.Add(Me.m_OtherSettingsGroupBox)
1176        Me.m_OtherTabPage.Location = New System.Drawing.Point(4, 22)
1177        Me.m_OtherTabPage.Name = "m_OtherTabPage"
1178        Me.m_OtherTabPage.Size = New System.Drawing.Size(630, 238)
1179        Me.m_OtherTabPage.TabIndex = 2
1180        Me.m_OtherTabPage.Text = "Other"
1181        '
1182        'm_OtherSettingsGroupBox
1183        '
1184        Me.m_OtherSettingsGroupBox.Controls.Add(Me.m_EnableProfilingMemoryCheckBox)
1185        Me.m_OtherSettingsGroupBox.Controls.Add(Me.m_EnableProfilingCheckBox)
1186        Me.m_OtherSettingsGroupBox.Controls.Add(Me.m_ThreeFileInputCheckBox)
1187        Me.m_OtherSettingsGroupBox.Controls.Add(Me.m_SingleFileInputCheckBox)
1188        Me.m_OtherSettingsGroupBox.Controls.Add(Me.m_RunEffectsModelCheckBox)
1189        Me.m_OtherSettingsGroupBox.Controls.Add(Me.m_NoStartupWarningCheckBox)
1190        Me.m_OtherSettingsGroupBox.Location = New System.Drawing.Point(8, 8)
1191        Me.m_OtherSettingsGroupBox.Name = "m_OtherSettingsGroupBox"
1192        Me.m_OtherSettingsGroupBox.Size = New System.Drawing.Size(616, 216)
1193        Me.m_OtherSettingsGroupBox.TabIndex = 15
1194        Me.m_OtherSettingsGroupBox.TabStop = False
1195        Me.m_OtherSettingsGroupBox.Text = "Other Settings"
1196        '
1197        'm_EnableProfilingMemoryCheckBox
1198        '
1199        Me.m_EnableProfilingMemoryCheckBox.Location = New System.Drawing.Point(32, 72)
1200        Me.m_EnableProfilingMemoryCheckBox.Name = "m_EnableProfilingMemoryCheckBox"
1201        Me.m_EnableProfilingMemoryCheckBox.Size = New System.Drawing.Size(192, 16)
1202        Me.m_EnableProfilingMemoryCheckBox.TabIndex = 2
1203        Me.m_EnableProfilingMemoryCheckBox.Text = "Track memory usage"
1204        '
1205        'm_EnableProfilingCheckBox
1206        '
1207        Me.m_EnableProfilingCheckBox.Location = New System.Drawing.Point(16, 48)
1208        Me.m_EnableProfilingCheckBox.Name = "m_EnableProfilingCheckBox"
1209        Me.m_EnableProfilingCheckBox.Size = New System.Drawing.Size(208, 16)
1210        Me.m_EnableProfilingCheckBox.TabIndex = 1
1211        Me.m_EnableProfilingCheckBox.Text = "Enable model performance profiling"
1212        '
1213        'm_ThreeFileInputCheckBox
1214        '
1215        Me.m_ThreeFileInputCheckBox.Location = New System.Drawing.Point(16, 144)
1216        Me.m_ThreeFileInputCheckBox.Name = "m_ThreeFileInputCheckBox"
1217        Me.m_ThreeFileInputCheckBox.Size = New System.Drawing.Size(144, 16)
1218        Me.m_ThreeFileInputCheckBox.TabIndex = 5
1219        Me.m_ThreeFileInputCheckBox.Text = "Use 3-file dataset input"
1220        Me.m_ThreeFileInputCheckBox.Visible = False
1221        '
1222        'm_SingleFileInputCheckBox
1223        '
1224        Me.m_SingleFileInputCheckBox.Location = New System.Drawing.Point(16, 120)
1225        Me.m_SingleFileInputCheckBox.Name = "m_SingleFileInputCheckBox"
1226        Me.m_SingleFileInputCheckBox.Size = New System.Drawing.Size(168, 16)
1227        Me.m_SingleFileInputCheckBox.TabIndex = 4
1228        Me.m_SingleFileInputCheckBox.Text = "Use Single-file dataset input"
1229        Me.m_SingleFileInputCheckBox.Visible = False
1230        '
1231        'm_RunEffectsModelCheckBox
1232        '
1233        Me.m_RunEffectsModelCheckBox.Location = New System.Drawing.Point(16, 96)
```

```
1234        Me.m_RunEffectsModelCheckBox.Name = "m_RunEffectsModelCheckBox"
1235        Me.m_RunEffectsModelCheckBox.Size = New System.Drawing.Size(280, 16)
1236        Me.m_RunEffectsModelCheckBox.TabIndex = 3
1237        Me.m_RunEffectsModelCheckBox.Text = "Run the Effects Model at the end of each
      scenario"
1238        Me.m_RunEffectsModelCheckBox.Visible = False
1239        '
1240        'm_NoStartupWarningCheckBox
1241        '
1242        Me.m_NoStartupWarningCheckBox.Location = New System.Drawing.Point(16, 24)
1243        Me.m_NoStartupWarningCheckBox.Name = "m_NoStartupWarningCheckBox"
1244        Me.m_NoStartupWarningCheckBox.Size = New System.Drawing.Size(208, 16)
1245        Me.m_NoStartupWarningCheckBox.TabIndex = 0
1246        Me.m_NoStartupWarningCheckBox.Text = "Do not display the warning at startup"
1247        '
1248        'm_OtherTemplatesGroupBox
1249        '
1250        Me.m_OtherTemplatesGroupBox.Controls.Add(Me.Label1)
1251        Me.m_OtherTemplatesGroupBox.Location = New System.Drawing.Point(320, 8)
1252        Me.m_OtherTemplatesGroupBox.Name = "m_OtherTemplatesGroupBox"
1253        Me.m_OtherTemplatesGroupBox.Size = New System.Drawing.Size(304, 216)
1254        Me.m_OtherTemplatesGroupBox.TabIndex = 1
1255        Me.m_OtherTemplatesGroupBox.TabStop = False
1256        Me.m_OtherTemplatesGroupBox.Text = "Other Templates"
1257        '
1258        'Label1
1259        '
1260        Me.Label1.Location = New System.Drawing.Point(104, 104)
1261        Me.Label1.Name = "Label1"
1262        Me.Label1.Size = New System.Drawing.Size(100, 16)
1263        Me.Label1.TabIndex = 0
1264        Me.Label1.Text = "(not implemented)"
1265        '
1266        'm_OutputTemplatesGroupBox
1267        '
1268        Me.m_OutputTemplatesGroupBox.Controls.Add(Me.m_UseOutputTemplatesPanel)
1269        Me.m_OutputTemplatesGroupBox.Controls.Add(Me.m_UseOutputTemplatesCheckBox2)
1270        Me.m_OutputTemplatesGroupBox.Location = New System.Drawing.Point(8, 8)
1271        Me.m_OutputTemplatesGroupBox.Name = "m_OutputTemplatesGroupBox"
1272        Me.m_OutputTemplatesGroupBox.Size = New System.Drawing.Size(304, 216)
1273        Me.m_OutputTemplatesGroupBox.TabIndex = 0
1274        Me.m_OutputTemplatesGroupBox.TabStop = False
1275        Me.m_OutputTemplatesGroupBox.Text = "Output Templates"
1276        '
1277        'm_UseOutputTemplatesPanel
1278        '
1279        Me.m_UseOutputTemplatesPanel.Controls.Add(Me.m_PortraitOutputTemplateBrowseButton)
1280        Me.m_UseOutputTemplatesPanel.Controls.Add(Me.m_PortraitOutputTemplateTextBox)
1281        Me.m_UseOutputTemplatesPanel.Controls.Add(Me.m_PortraitOutputTemplateLabel)
1282        Me.m_UseOutputTemplatesPanel.Controls.Add(Me.
      m_LandscapeOutputTemplateBrowseButton)
1283        Me.m_UseOutputTemplatesPanel.Controls.Add(Me.m_LandscapeOutputTemplateTextBox)
1284        Me.m_UseOutputTemplatesPanel.Controls.Add(Me.m_LandscapeOutputTemplateLabel)
1285        Me.m_UseOutputTemplatesPanel.Controls.Add(Me.m_DefaultOutputTemplateBrowseButton2)
1286        Me.m_UseOutputTemplatesPanel.Controls.Add(Me.m_DefaultOutputTemplateTextBox2)
1287        Me.m_UseOutputTemplatesPanel.Controls.Add(Me.m_DefaultOutputTemplateLabel)
1288        Me.m_UseOutputTemplatesPanel.Location = New System.Drawing.Point(8, 40)
1289        Me.m_UseOutputTemplatesPanel.Name = "m_UseOutputTemplatesPanel"
1290        Me.m_UseOutputTemplatesPanel.Size = New System.Drawing.Size(288, 168)
1291        Me.m_UseOutputTemplatesPanel.TabIndex = 1
1292        '
1293        'm_PortraitOutputTemplateBrowseButton
1294        '
1295        Me.m_PortraitOutputTemplateBrowseButton.Location = New System.Drawing.Point(200,
      144)
1296        Me.m_PortraitOutputTemplateBrowseButton.Name =
      "m_PortraitOutputTemplateBrowseButton"
1297        Me.m_PortraitOutputTemplateBrowseButton.Size = New System.Drawing.Size(80, 20)
1298        Me.m_PortraitOutputTemplateBrowseButton.TabIndex = 8
1299        Me.m_PortraitOutputTemplateBrowseButton.Text = "Browse..."
```

```
1300        '
1301        'm_PortraitOutputTemplateTextBox
1302        '
1303        Me.m_PortraitOutputTemplateTextBox.BackColor = System.Drawing.SystemColors.Window
1304        Me.m_PortraitOutputTemplateTextBox.Location = New System.Drawing.Point(8, 120)
1305        Me.m_PortraitOutputTemplateTextBox.Name = "m_PortraitOutputTemplateTextBox"
1306        Me.m_PortraitOutputTemplateTextBox.ReadOnly = True
1307        Me.m_PortraitOutputTemplateTextBox.Size = New System.Drawing.Size(272, 20)
1308        Me.m_PortraitOutputTemplateTextBox.TabIndex = 7
1309        Me.m_PortraitOutputTemplateTextBox.Text = ""
1310        '
1311        'm_PortraitOutputTemplateLabel
1312        '
1313        Me.m_PortraitOutputTemplateLabel.Location = New System.Drawing.Point(8, 104)
1314        Me.m_PortraitOutputTemplateLabel.Name = "m_PortraitOutputTemplateLabel"
1315        Me.m_PortraitOutputTemplateLabel.Size = New System.Drawing.Size(184, 16)
1316        Me.m_PortraitOutputTemplateLabel.TabIndex = 6
1317        Me.m_PortraitOutputTemplateLabel.Text = """Portrait"" Template"
1318        '
1319        'm_LandscapeOutputTemplateBrowseButton
1320        '
1321        Me.m_LandscapeOutputTemplateBrowseButton.Location = New System.Drawing.Point(200,
    96)
1322        Me.m_LandscapeOutputTemplateBrowseButton.Name =
    "m_LandscapeOutputTemplateBrowseButton"
1323        Me.m_LandscapeOutputTemplateBrowseButton.Size = New System.Drawing.Size(80, 20)
1324        Me.m_LandscapeOutputTemplateBrowseButton.TabIndex = 5
1325        Me.m_LandscapeOutputTemplateBrowseButton.Text = "Browse..."
1326        '
1327        'm_LandscapeOutputTemplateTextBox
1328        '
1329        Me.m_LandscapeOutputTemplateTextBox.BackColor = System.Drawing.SystemColors.Window
1330        Me.m_LandscapeOutputTemplateTextBox.Location = New System.Drawing.Point(8, 72)
1331        Me.m_LandscapeOutputTemplateTextBox.Name = "m_LandscapeOutputTemplateTextBox"
1332        Me.m_LandscapeOutputTemplateTextBox.ReadOnly = True
1333        Me.m_LandscapeOutputTemplateTextBox.Size = New System.Drawing.Size(272, 20)
1334        Me.m_LandscapeOutputTemplateTextBox.TabIndex = 4
1335        Me.m_LandscapeOutputTemplateTextBox.Text = ""
1336        '
1337        'm_LandscapeOutputTemplateLabel
1338        '
1339        Me.m_LandscapeOutputTemplateLabel.Location = New System.Drawing.Point(8, 56)
1340        Me.m_LandscapeOutputTemplateLabel.Name = "m_LandscapeOutputTemplateLabel"
1341        Me.m_LandscapeOutputTemplateLabel.Size = New System.Drawing.Size(184, 16)
1342        Me.m_LandscapeOutputTemplateLabel.TabIndex = 3
1343        Me.m_LandscapeOutputTemplateLabel.Text = """Landscape"" Template"
1344        '
1345        'm_DefaultOutputTemplateBrowseButton2
1346        '
1347        Me.m_DefaultOutputTemplateBrowseButton2.Location = New System.Drawing.Point(200,
    48)
1348        Me.m_DefaultOutputTemplateBrowseButton2.Name =
    "m_DefaultOutputTemplateBrowseButton2"
1349        Me.m_DefaultOutputTemplateBrowseButton2.Size = New System.Drawing.Size(80, 20)
1350        Me.m_DefaultOutputTemplateBrowseButton2.TabIndex = 2
1351        Me.m_DefaultOutputTemplateBrowseButton2.Text = "Browse..."
1352        '
1353        'm_DefaultOutputTemplateTextBox2
1354        '
1355        Me.m_DefaultOutputTemplateTextBox2.BackColor = System.Drawing.SystemColors.Window
1356        Me.m_DefaultOutputTemplateTextBox2.Location = New System.Drawing.Point(8, 24)
1357        Me.m_DefaultOutputTemplateTextBox2.Name = "m_DefaultOutputTemplateTextBox2"
1358        Me.m_DefaultOutputTemplateTextBox2.ReadOnly = True
1359        Me.m_DefaultOutputTemplateTextBox2.Size = New System.Drawing.Size(272, 20)
1360        Me.m_DefaultOutputTemplateTextBox2.TabIndex = 1
1361        Me.m_DefaultOutputTemplateTextBox2.Text = ""
1362        '
1363        'm_DefaultOutputTemplateLabel
1364        '
1365        Me.m_DefaultOutputTemplateLabel.Location = New System.Drawing.Point(8, 8)
```

```
1366            Me.m_DefaultOutputTemplateLabel.Name = "m_DefaultOutputTemplateLabel"
1367            Me.m_DefaultOutputTemplateLabel.Size = New System.Drawing.Size(272, 16)
1368            Me.m_DefaultOutputTemplateLabel.TabIndex = 0
1369            Me.m_DefaultOutputTemplateLabel.Text = "Default Output Template"
1370            '
1371            'm_UseOutputTemplatesCheckBox2
1372            '
1373            Me.m_UseOutputTemplatesCheckBox2.Location = New System.Drawing.Point(16, 24)
1374            Me.m_UseOutputTemplatesCheckBox2.Name = "m_UseOutputTemplatesCheckBox2"
1375            Me.m_UseOutputTemplatesCheckBox2.Size = New System.Drawing.Size(224, 16)
1376            Me.m_UseOutputTemplatesCheckBox2.TabIndex = 0
1377            Me.m_UseOutputTemplatesCheckBox2.Text = "Use Templates For Generating Outputs"
1378            '
1379            'OptionsForm
1380            '
1381            Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
1382            Me.ClientSize = New System.Drawing.Size(638, 323)
1383            Me.ControlBox = False
1384            Me.Controls.Add(Me.m_OptionsTabControl)
1385            Me.Controls.Add(Me.m_ApplyButton)
1386            Me.Controls.Add(Me.m_CancelButton)
1387            Me.Controls.Add(Me.m_OKButton)
1388            Me.Controls.Add(Me.m_ModelYearsButton)
1389            Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D
1390            Me.MaximizeBox = False
1391            Me.MinimizeBox = False
1392            Me.Name = "OptionsForm"
1393            Me.Text = "Options"
1394            Me.m_PathsGroupBox.ResumeLayout(False)
1395            Me.m_FuelGroupBox.ResumeLayout(False)
1396            Me.m_CostGroupBox.ResumeLayout(False)
1397            Me.m_OptionsTabControl.ResumeLayout(False)
1398            Me.m_TechSettingsTabPage.ResumeLayout(False)
1399            Me.m_OperatingModesTabPage.ResumeLayout(False)
1400            Me.m_OperatingModesGroupBox.ResumeLayout(False)
1401            CType(Me.m_MonteCarloIterationsNumericUpDown, System.ComponentModel.
        ISupportInitialize).EndInit()
1402            Me.m_EncryptionSettingsTabPage.ResumeLayout(False)
1403            Me.m_EncryptionSettingsSavingGroupBox.ResumeLayout(False)
1404            Me.m_EncryptOutputFilesPanel.ResumeLayout(False)
1405            CType(Me.m_KeyLengthNumericUpDown, System.ComponentModel.ISupportInitialize).
        EndInit()
1406            Me.m_EncryptionSettingsLoadingGroupBox.ResumeLayout(False)
1407            Me.m_ReadEncryptedFilesPanel.ResumeLayout(False)
1408            Me.m_OutputSettingsTabPage.ResumeLayout(False)
1409            Me.m_TemplateSettingsGroupBox.ResumeLayout(False)
1410            Me.m_ReportSettingsGroupBox.ResumeLayout(False)
1411            Me.m_GeneralGroupBox.ResumeLayout(False)
1412            Me.m_OtherTabPage.ResumeLayout(False)
1413            Me.m_OtherSettingsGroupBox.ResumeLayout(False)
1414            Me.m_OtherTemplatesGroupBox.ResumeLayout(False)
1415            Me.m_OutputTemplatesGroupBox.ResumeLayout(False)
1416            Me.m_UseOutputTemplatesPanel.ResumeLayout(False)
1417            Me.ResumeLayout(False)
1418
1419        End Sub
1420
1421    #End Region
1422
1423        Private Sub OnOptionsFormIdle(ByVal sender As System.Object, ByVal e As System.
        EventArgs)
1424            ' enable or disable the apply button based on whether or not
1425            ' any changes have been made on the options form since loading
1426            Me.m_ApplyButton.Enabled = Me.ChangeDetected
1427
1428            ' operating modes tab
1429            Me.m_SelectAllModesCheckBox.CheckState = Me.GetOperatingModesCheckedState()
1430            Me.m_CreateEngineIfAllModesCheckBox.CheckState = Me.
        GetCreateEngineModesCheckedState()
1431            Me.m_CreateTransmissionIfAllModesCheckBox.CheckState = Me.
```

```
    GetCreateTransmissionModesCheckedState()
1432        Me.m_SkipEngineTransmissionIfAllModesCheckBox.CheckState = Me.
    GetSkipEngineTransmissionModesCheckedState()
1433        Me.m_MonteCarloIterationsNumericUpDown.Enabled = Me.m_MonteCarloAnalysisCheckBox.
    Checked
1434
1435        ' output settings tab
1436        Me.m_ConfigureTemplatesButton.Enabled = Me.m_UseOutputTemplatesCheckBox.Checked
1437        '
1438        If Me.m_GenIndustryReportCheckBox.Checked AndAlso Me.m_GenMfgsReportCheckBox.
    Checked AndAlso _
1439            Me.m_GenVehiclesReportCheckBox.Checked AndAlso Me.m_GenEffectsReportCheckBox.
    Checked AndAlso _
1440            Me.m_GenEAReportCheckBox.Checked Then
1441            '
1442            Me.m_GenAllReportsCheckBox.CheckState = CheckState.Checked
1443        ElseIf Me.m_GenIndustryReportCheckBox.Checked OrElse Me.m_GenMfgsReportCheckBox.
    Checked OrElse _
1444                Me.m_GenVehiclesReportCheckBox.Checked OrElse Me.m_GenEffectsReportCheckBox
    .Checked OrElse _
1445                Me.m_GenEAReportCheckBox.Checked Then
1446            '
1447            Me.m_GenAllReportsCheckBox.CheckState = CheckState.Indeterminate
1448        Else
1449            Me.m_GenAllReportsCheckBox.CheckState = CheckState.Unchecked
1450        End If
1451        ' enable effects reporting only if run effects model option is selected
1452        Me.m_GenEffectsReportCheckBox.Enabled = Me.m_RunEffectsModelCheckBox.Checked
1453        ' uncheck effects report option
1454        If Not Me.m_GenEffectsReportCheckBox.Enabled Then
1455            Me.m_GenEffectsReportCheckBox.Checked = False
1456        End If
1457        ' enable/disable template path textbox and template browse button
1458        Me.m_DefaultOutputTemplateTextBox.Enabled = Me.m_UseOutputTemplatesCheckBox.
    Checked
1459        Me.m_DefaultOutputTemplateBrowseButton.Enabled = Me.m_UseOutputTemplatesCheckBox.
    Checked
1460
1461        ' template settings
1462        Me.m_UseOutputTemplatesPanel.Enabled = Me.m_UseOutputTemplatesCheckBox2.Checked
1463
1464        ' other settings
1465        Me.m_EnableProfilingMemoryCheckBox.Enabled = Me.m_EnableProfilingCheckBox.Checked
1466    End Sub
1467
1468    Private Sub OnOptionsFormLoad(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles MyBase.Load
1469        Dim ModelRunning As Boolean = CAFE.ModelRunning
1470        Dim i As Integer
1471
1472        AddHandler Application.Idle, AddressOf OnOptionsFormIdle
1473
1474        ' initialize the tooltip
1475        Me.MyToolTip = New System.Windows.Forms.ToolTip
1476        Me.MyToolTip.Active = True
1477        Me.MyToolTip.InitialDelay = 100
1478        Me.MyToolTip.ReshowDelay = 100
1479        Me.MyToolTip.AutoPopDelay = 25000
1480
1481        m_ModelYearsButton.Enabled = Not IsNothing(CAFE.ModelYearsSelected)
1482
1483        ' set whether the tab pages are enabled
1484        If CAFE.ModelRunning Then
1485            Me.m_TechSettingsTabPage.Enabled = False
1486            Me.m_OperatingModesTabPage.Enabled = False
1487            Me.m_EncryptionSettingsTabPage.Enabled = False
1488            Me.m_OutputSettingsTabPage.Enabled = False
1489            'Me.m_TemplatesTabPage.Enabled = False
1490            Me.m_OtherTabPage.Enabled = False
1491        End If
```

```
1492
1493            ' set technology paths
1494            Me.m_GroupBoxPath1.Checked = (CAFE.PathSettings = CAFE.PathsEnum.Path1)
1495            Me.m_GroupBoxPath2.Checked = (CAFE.PathSettings = CAFE.PathsEnum.Path2)
1496            Me.m_GroupBoxPath3.Checked = (CAFE.PathSettings = CAFE.PathsEnum.Path3)
1497            Me.PathSettings = CAFE.PathSettings
1498
1499            ' set technology fuel consumption estimates
1500            Me.m_GroupBoxFuelLow.Checked = (CAFE.FuelConsumptionSavingsEstimates = CAFE.
        LowAverageHigh.Low)
1501            Me.m_GroupBoxFuelAverage.Checked = (CAFE.FuelConsumptionSavingsEstimates = CAFE.
        LowAverageHigh.Average)
1502            Me.m_GroupBoxFuelHigh.Checked = (CAFE.FuelConsumptionSavingsEstimates = CAFE.
        LowAverageHigh.High)
1503            FuelConsumptionSavingsEstimates = CAFE.FuelConsumptionSavingsEstimates
1504
1505            ' set technology cost estimates
1506            Me.m_GroupBoxCostLow.Checked = (CAFE.TechnologyCostEstimates = CAFE.LowAverageHigh
        .Low)
1507            Me.m_GroupBoxCostAverage.Checked = (CAFE.TechnologyCostEstimates = CAFE.
        LowAverageHigh.Average)
1508            Me.m_GroupBoxCostHigh.Checked = (CAFE.TechnologyCostEstimates = CAFE.
        LowAverageHigh.High)
1509            Me.TechnologyCostEstimates = CAFE.TechnologyCostEstimates
1510
1511            ' set operating modes
1512            With CAFE.OperatingModes
1513                Me.m_NoFinesModeCheckBox.Checked = .NoFines
1514                Me.m_LowCostTechnologyFirstModeCheckBox.Checked = (.Approach = CAFE.
        ApproachEnum.LowCostFirst)
1515                Me.m_AutoInvokeEfficiencyModeCheckBox.Checked = .AutoToggleApproach
1516                Me.m_BackFillTechnologyModeCheckBox.Checked = .BackFillTechnology
1517                Me.m_StageAnalysisCheckBox.Checked = .StageAnalysis
1518                Me.m_MonteCarloAnalysisCheckBox.Checked = .MonteCarloAnalysis
1519                Me.m_MonteCarloIterationsNumericUpDown.Value = .MonteCarloIterations
1520                Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Checked = .
        CreateEngineIfCarsInvolved
1521                Me.m_CreateEngineIfMultipleRCModeCheckBox.Checked = .CreateEngineIfMultipleRC
1522                Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Checked = .
        CreateEngineIfTechEnableMissMatch
1523                Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Checked = .
        CreateTransmissionIfCarsInvolved
1524                Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Checked = .
        CreateTransmissionIfMultipleRC
1525                Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Checked = .
        CreateTransmissionIfTechEnableMissMatch
1526                Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Checked = .
        SkipIfAutoLightTruckMix
1527                Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Checked = .SkipIfMultiRC
1528                Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Checked = .
        SkipIfEnableMismatch
1529                Me.m_ReserveTechsIfInComplianceModeCheckBox.Checked = .
        ReserveTechsIfInCompliance
1530            End With
1531
1532            ' set encryption settings
1533            With CAFE.EncryptionSettings
1534                ' set file saving settings
1535                Me.m_EncryptOutputFilesCheckBox.Checked = .EncryptOutputFiles
1536                Me.m_EncryptOutputFilesPanel.Enabled = Me.m_EncryptOutputFilesCheckBox.Checked
1537                For i = 1 To CAFE.PasswordEncryptionProviders.Length - 1
1538                    Me.m_EncryptionProviderComboBox.Items.Add(CAFE.PasswordEncryptionProviders
        (i))
1539                Next i
1540                Me.m_EncryptionProviderComboBox.Text = .EncryptionProvider
1541                If .EncryptionKeyLength < 64 Or _
1542                   .EncryptionKeyLength > 128 Then
1543                    Me.m_KeyLengthNumericUpDown.Text = 128
1544                Else
1545                    Me.m_KeyLengthNumericUpDown.Text = .EncryptionKeyLength
```

```
1546            End If
1547            Me.m_EncryptDocPropertiesCheckBox.Checked = .EncryptDocumentProperties
1548            Me.m_PasswordToOpenTextBox.Text = .OutputPasswordToOpen
1549            If .OutputPasswordToOpen = .OutputPasswordToModify Then
1550                Me.m_PasswordToModifyCheckBox.Checked = True
1551            Else
1552                Me.m_PasswordToModifyTextBox.Text = .OutputPasswordToModify
1553            End If
1554            ' set file loading settings
1555            Me.m_ReadEncryptedFilesCheckBox.Checked = .ReadEncryptedInputFiles
1556            If .UseMultiplePasswords Then
1557                Me.m_InputSinglePwCheckBox.Checked = False
1558                Me.m_InputMultiPwCheckBox.Checked = True
1559                Me.m_InputMultiPwVehTextBox.Text = .InputVehPassword
1560                Me.m_InputMultiPwEngTextBox.Text = .InputEngPassword
1561                Me.m_InputMultiPwTranTextBox.Text = .InputTranPassword
1562                Me.m_InputMultiPwTechTextBox.Text = .InputTechPassword
1563            Else
1564                Me.m_InputSinglePwCheckBox.Checked = True
1565                Me.m_InputMultiPwCheckBox.Checked = False
1566                Me.m_InputSinglePwTextBox.Text = .InputGlobalPassword
1567            End If
1568        End With
1569
1570        Me.m_ReadEncryptedFilesPanel.Enabled = Me.m_ReadEncryptedFilesCheckBox.Checked
1571
1572        ' set output settings
1573        With CAFE.OutputSettings
1574            Me.m_DefaultOutputPathTextBox.Text = .DefaultOutputPath
1575            Me.m_HideUnusedRCsCheckBox.Checked = .HideUnusedRCs
1576            ' default template
1577            Me.m_DefaultOutputTemplateTextBox.Text = CAFE.TemplateSettings.
    DefaultTemplatePath
1578            ' report settings
1579            Me.m_GenIndustryReportCheckBox.Checked = .GenerateIndustryReport
1580            Me.m_GenMfgsReportCheckBox.Checked = .GenerateMfgsReport
1581            Me.m_GenVehiclesReportCheckBox.Checked = .GenerateVehiclesReport
1582            Me.m_GenEffectsReportCheckBox.Checked = .GenerateEffectsReport
1583            Me.m_GenEAReportCheckBox.Checked = .GenerateEnvironmentalAssessmentReport
1584        End With
1585
1586        ' set template settings
1587        With CAFE.TemplateSettings
1588            Me.m_UseOutputTemplatesCheckBox2.Checked = .UseTemplates
1589            Me.m_DefaultOutputTemplateTextBox2.Text = .DefaultTemplatePath
1590
1591            ' for now, disable portrait/landscape templates
1592            Me.m_LandscapeOutputTemplateLabel.Enabled = False
1593            Me.m_LandscapeOutputTemplateTextBox.TabStop = False
1594            Me.m_LandscapeOutputTemplateTextBox.ForeColor = Color.Gray
1595            Me.m_LandscapeOutputTemplateBrowseButton.Enabled = False
1596            Me.m_PortraitOutputTemplateLabel.Enabled = False
1597            Me.m_PortraitOutputTemplateTextBox.TabStop = False
1598            Me.m_PortraitOutputTemplateTextBox.ForeColor = Color.Gray
1599            Me.m_PortraitOutputTemplateBrowseButton.Enabled = False
1600
1601            Me.MyToolTip.SetToolTip(Me.m_LandscapeOutputTemplateLabel, " This option is
    not yet available. ")
1602            Me.MyToolTip.SetToolTip(Me.m_LandscapeOutputTemplateTextBox, " This option is
    not yet available. ")
1603            Me.MyToolTip.SetToolTip(Me.m_LandscapeOutputTemplateBrowseButton, " This
    option is not yet available. ")
1604            Me.MyToolTip.SetToolTip(Me.m_PortraitOutputTemplateLabel, " This option is not
     yet available. ")
1605            Me.MyToolTip.SetToolTip(Me.m_PortraitOutputTemplateTextBox, " This option is
    not yet available. ")
1606            Me.MyToolTip.SetToolTip(Me.m_PortraitOutputTemplateBrowseButton, " This option
     is not yet available. ")
1607        End With
1608
```

```
1609        ' set other settings
1610        With CAFE.OtherSettings
1611            Me.m_RunEffectsModelCheckBox.Checked = .RunEffectsModel
1612            Me.m_NoStartupWarningCheckBox.Checked = .NoStartupWarning
1613            Me.m_SingleFileInputCheckBox.Checked = .SingleFileDatasetInput
1614            Me.m_ThreeFileInputCheckBox.Checked = Not .SingleFileDatasetInput
1615        End With
1616        ' profiling
1617        With CAFE.Profiling
1618            Me.m_EnableProfilingCheckBox.Checked = .DEBUG
1619            Me.m_EnableProfilingMemoryCheckBox.Checked = .GET_MEM_USAGE
1620
1621            Me.MyToolTip.SetToolTip(Me.m_EnableProfilingCheckBox, _
1622                    " This option enables model profiling in order to generate model performance " & NewLine & _
1623                    " logs of individual subs and functions for debugging purposes. Profiling is not " & NewLine & _
1624                    " available for all methods and only the selected ones are profiled in several " & NewLine & _
1625                    " classes.Current classes that support profiling are: xlUtilities (Module), " & NewLine & _
1626                    " ManufacturerClass, MarketSimulationClass, ModellingLogic, and Output. ")
1627            Me.MyToolTip.SetToolTip(Me.m_EnableProfilingMemoryCheckBox, _
1628                    " By default, only runtimes for each method are tracked during profiling. Turn " & NewLine & _
1629                    " this option on to enable memory and VRAM tracking as well. Tracking memory " & NewLine & _
1630                    " will result in slower model run times, leading to runtimes being longer than " & NewLine & _
1631                    " they really are. ")
1632        End With
1633
1634        Me.ChangeDetected = False       ' no changes have been made, since the form just loaded
1635    End Sub
1636
1637    Private Sub OnModelYearsButton(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles m_ModelYearsButton.Click
1638        Debug.WriteLine("Model Year Settings Menu")
1639        Dim MYSettings As MYForm = New MYForm(Me, 0)
1640        MYSettings.ShowDialog()
1641    End Sub
1642
1643    Private Function FolderBrowserDialogHelper(ByRef Control As System.Windows.Forms.TextBox, _
1644                                               ByVal Description As String, _
1645                                               Optional ByVal RootFolder As Environment.SpecialFolder = Environment.SpecialFolder.Desktop) As Boolean
1646        '
1647        Me.m_FolderBrowserDialog.Description = Description
1648        Me.m_FolderBrowserDialog.RootFolder = RootFolder
1649        If Control.Text <> "" Then
1650            Me.m_FolderBrowserDialog.SelectedPath = Control.Text
1651        Else
1652            Me.m_FolderBrowserDialog.SelectedPath = RootFolder.ToString()
1653        End If
1654
1655        If Me.m_FolderBrowserDialog.ShowDialog() = DialogResult.OK Then
1656            Control.Text = Me.m_FolderBrowserDialog.SelectedPath
1657            Return True
1658        End If
1659
1660        Return False
1661    End Function
1662
1663    Private Function FileBrowserDialogHelper(ByRef Control As System.Windows.Forms.TextBox, _
1664                                             ByVal Description As String) As Boolean
1665        '
```

```
1666            Me.m_OpenFileDialog.Title = Description
1667            Me.m_OpenFileDialog.CheckFileExists = True
1668            Me.m_OpenFileDialog.Multiselect = False
1669            Me.m_OpenFileDialog.Filter = "Excel Files (*.xls)|*.xls"
1670            Me.m_OpenFileDialog.FilterIndex = 1
1671            Me.m_OpenFileDialog.InitialDirectory = Control.Text
1672
1673            If Me.m_OpenFileDialog.ShowDialog() = DialogResult.OK Then
1674                Control.Text = Me.m_OpenFileDialog.FileName
1675                Return True
1676            Else
1677                Return False
1678            End If
1679        End Function
1680
1681
1682    #Region " Apply - OK - Cancel buttons "
1683
1684        Private Sub OnApplyButton(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles m_ApplyButton.Click
1685            ' save options selections to CAFE object
1686            CAFE.PathSettings = PathSettings
1687            CAFE.FuelConsumptionSavingsEstimates = FuelConsumptionSavingsEstimates
1688            CAFE.TechnologyCostEstimates = TechnologyCostEstimates
1689
1690            ' save operating modes
1691            With CAFE.OperatingModes
1692                .NoFines = Me.m_NoFinesModeCheckBox.Checked
1693                If Me.m_LowCostTechnologyFirstModeCheckBox.Checked Then
1694                    .Approach = CAFE.ApproachEnum.LowCostFirst
1695                Else
1696                    .Approach = CAFE.ApproachEnum.EconomicEfficiency
1697                End If
1698                .AutoToggleApproach = Me.m_AutoInvokeEfficiencyModeCheckBox.Checked
1699                .BackFillTechnology = Me.m_BackFillTechnologyModeCheckBox.Checked
1700                .StageAnalysis = Me.m_StageAnalysisCheckBox.Checked
1701                .MonteCarloAnalysis = Me.m_MonteCarloAnalysisCheckBox.Checked
1702                .MonteCarloIterations = CInt(Me.m_MonteCarloIterationsNumericUpDown.Value)
1703                .CreateEngineIfCarsInvolved = Me.m_CreateEngineIfCarsInvolvedModeCheckBox.
        Checked
1704                .CreateEngineIfMultipleRC = Me.m_CreateEngineIfMultipleRCModeCheckBox.Checked
1705                .CreateEngineIfTechEnableMissMatch = Me.
        m_CreateEngineIfTechEnableMissMatchModeCheckBox.Checked
1706                .CreateTransmissionIfCarsInvolved = Me.
        m_CreateTransmissionIfCarsInvolvedModeCheckBox.Checked
1707                .CreateTransmissionIfMultipleRC = Me.
        m_CreateTransmissionIfMultipleRCModeCheckBox.Checked
1708                .CreateTransmissionIfTechEnableMissMatch = Me.
        m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Checked
1709                .SkipIfAutoLightTruckMix = Me.
        m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Checked
1710                .SkipIfMultiRC = Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Checked
1711                .SkipIfEnableMismatch = Me.
        m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Checked
1712                .ReserveTechsIfInCompliance = Me.m_ReserveTechsIfInComplianceModeCheckBox.
        Checked
1713            End With
1714
1715            ' save encryption settings
1716            With CAFE.EncryptionSettings
1717                ' file saving settings
1718                If Me.m_EncryptOutputFilesCheckBox.Checked Then
1719                    .EncryptOutputFiles = True
1720                    .EncryptionProvider = Me.m_EncryptionProviderComboBox.Text
1721                    .EncryptionKeyLength = Me.m_KeyLengthNumericUpDown.Value
1722                    If .EncryptionKeyLength < 64 Then
1723                        .EncryptionKeyLength = 64
1724                    ElseIf .EncryptionKeyLength > 128 Then
1725                        .EncryptionKeyLength = 128
1726                    End If
```

```
1727                .EncryptDocumentProperties = Me.m_EncryptDocPropertiesCheckBox.Checked
1728                .OutputPasswordToOpen = Me.m_PasswordToOpenTextBox.Text
1729                .OutputPasswordToModify = Me.m_PasswordToModifyTextBox.Text
1730            Else
1731                .EncryptOutputFiles = False
1732            End If
1733            ' file loading settings
1734            If Me.m_ReadEncryptedFilesCheckBox.Checked Then
1735                .ReadEncryptedInputFiles = True
1736                .UseMultiplePasswords = Me.m_InputMultiPwCheckBox.Checked
1737                If Me.m_InputSinglePwCheckBox.Checked Then
1738                    .InputGlobalPassword = Me.m_InputSinglePwTextBox.Text
1739                End If
1740                If Me.m_InputMultiPwCheckBox.Checked Then
1741                    .InputVehPassword = Me.m_InputMultiPwVehTextBox.Text
1742                    .InputEngPassword = Me.m_InputMultiPwEngTextBox.Text
1743                    .InputTranPassword = Me.m_InputMultiPwTranTextBox.Text
1744                    .InputTechPassword = Me.m_InputMultiPwTechTextBox.Text
1745                End If
1746            Else
1747                .ReadEncryptedInputFiles = False
1748            End If
1749        End With
1750
1751        ' save output settings
1752        With CAFE.OutputSettings
1753            .DefaultOutputPath = Me.m_DefaultOutputPathTextBox.Text
1754            .HideUnusedRCs = Me.m_HideUnusedRCsCheckBox.Checked
1755            ' report settings
1756            .GenerateIndustryReport = Me.m_GenIndustryReportCheckBox.Checked
1757            .GenerateMfgsReport = Me.m_GenMfgsReportCheckBox.Checked
1758            .GenerateVehiclesReport = Me.m_GenVehiclesReportCheckBox.Checked
1759            .GenerateEffectsReport = Me.m_GenEffectsReportCheckBox.Checked
1760            .GenerateEnvironmentalAssessmentReport = Me.m_GenEAReportCheckBox.Checked
1761        End With
1762
1763        ' save template settings
1764        With CAFE.TemplateSettings
1765            .UseTemplates = Me.m_UseOutputTemplatesCheckBox2.Checked
1766            .DefaultTemplatePath = Me.m_DefaultOutputTemplateTextBox.Text
1767        End With
1768
1769        ' save other settings
1770        With CAFE.OtherSettings
1771            .RunEffectsModel = Me.m_RunEffectsModelCheckBox.Checked
1772            .NoStartupWarning = Me.m_NoStartupWarningCheckBox.Checked
1773            .SingleFileDatasetInput = Me.m_SingleFileInputCheckBox.Checked
1774        End With
1775        ' profiling
1776        With CAFE.Profiling
1777            .DEBUG = Me.m_EnableProfilingCheckBox.Checked
1778            .GET_MEM_USAGE = Me.m_EnableProfilingMemoryCheckBox.Checked
1779        End With
1780
1781        ' reset the change detected flag
1782        Me.ChangeDetected = False
1783    End Sub
1784
1785    Private Sub OnOKButton(ByVal sender As System.Object, ByVal e As System.EventArgs) ↙
        Handles m_OKButton.Click
1786        If Me.ChangeDetected Then
1787            OnApplyButton(sender, e)
1788        End If
1789        Me.Close()
1790    End Sub
1791
1792    Private Sub OnCancelButton(ByVal sender As System.Object, ByVal e As System.EventArgs) ↙
         Handles m_CancelButton.Click
1793        If Me.ChangeDetected Then
1794            If MsgBox("Some of the changes made have not been saved.  " & _
```

```
1795                        "Are you sure you want to cancel?",
1796                        MsgBoxStyle.YesNo, "Cancel changes?") = MsgBoxResult.Yes Then
1797                    Me.Close()
1798                Else
1799                    Exit Sub
1800                End If
1801            Else
1802                Me.Close()
1803            End If
1804        End Sub
1805
1806    #End Region
1807
1808
1809    #Region " Technology Settings "
1810
1811        Private Sub OnGroupBoxPath1_CheckedChanged(ByVal sender As System.Object, ByVal e As
            System.EventArgs) Handles m_GroupBoxPath1.CheckedChanged
1812            PathSettings = CAFE.PathsEnum.Path1
1813            Me.ChangeDetected = True
1814        End Sub
1815
1816        Private Sub OnGroupBoxPath2_CheckedChanged(ByVal sender As System.Object, ByVal e As
            System.EventArgs) Handles m_GroupBoxPath2.CheckedChanged
1817            PathSettings = CAFE.PathsEnum.Path2
1818            Me.ChangeDetected = True
1819        End Sub
1820
1821        Private Sub OnGroupBoxPath3_CheckedChanged(ByVal sender As System.Object, ByVal e As
            System.EventArgs) Handles m_GroupBoxPath3.CheckedChanged
1822            PathSettings = CAFE.PathsEnum.Path3
1823            Me.ChangeDetected = True
1824        End Sub
1825
1826        Private Sub OnGroupBoxFuelLow_CheckedChanged(ByVal sender As System.Object, ByVal e As
             System.EventArgs) Handles m_GroupBoxFuelLow.CheckedChanged
1827            FuelConsumptionSavingsEstimates = CAFE.LowAverageHigh.Low
1828            Me.ChangeDetected = True
1829        End Sub
1830
1831        Private Sub OnGroupBoxFuelAverage_CheckedChanged(ByVal sender As System.Object, ByVal
            e As System.EventArgs) Handles m_GroupBoxFuelAverage.CheckedChanged
1832            FuelConsumptionSavingsEstimates = CAFE.LowAverageHigh.Average
1833            Me.ChangeDetected = True
1834        End Sub
1835
1836        Private Sub OnGroupBoxFuelHigh_CheckedChanged(ByVal sender As System.Object, ByVal e
            As System.EventArgs) Handles m_GroupBoxFuelHigh.CheckedChanged
1837            FuelConsumptionSavingsEstimates = CAFE.LowAverageHigh.High
1838            Me.ChangeDetected = True
1839        End Sub
1840
1841        Private Sub OnGroupBoxCostLow_CheckedChanged(ByVal sender As System.Object, ByVal e As
             System.EventArgs) Handles m_GroupBoxCostLow.CheckedChanged
1842            TechnologyCostEstimates = CAFE.LowAverageHigh.Low
1843            Me.ChangeDetected = True
1844        End Sub
1845
1846        Private Sub OnGroupBoxCostAverage_CheckedChanged(ByVal sender As System.Object, ByVal
            e As System.EventArgs) Handles m_GroupBoxCostAverage.CheckedChanged
1847            TechnologyCostEstimates = CAFE.LowAverageHigh.Average
1848            Me.ChangeDetected = True
1849        End Sub
1850
1851        Private Sub OnGroupBoxCostHigh_CheckedChanged(ByVal sender As System.Object, ByVal e
            As System.EventArgs) Handles m_GroupBoxCostHigh.CheckedChanged
1852            TechnologyCostEstimates = CAFE.LowAverageHigh.High
1853            Me.ChangeDetected = True
1854        End Sub
1855
```

```
1856 #End Region
1857
1858
1859 #Region " Operating Modes "
1860
1861     Private Sub OnSelectAllModesCheckBox_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles m_SelectAllModesCheckBox.Click
1862         Me.m_SelectAllModesCheckBox.Checked = Not (Me.m_SelectAllModesCheckBox.CheckState
     = CheckState.Indeterminate)
1863
1864         If Me.m_SelectAllModesCheckBox.CheckState = CheckState.Checked Then
1865             Me.SetOperatingModes(True)
1866         ElseIf Me.m_SelectAllModesCheckBox.CheckState = CheckState.Unchecked Then
1867             Me.SetOperatingModes(False)
1868         End If
1869     End Sub
1870
1871     Private Sub OnCreateEngineIfAllModesCheckBox_Click(ByVal sender As System.Object,
     ByVal e As System.EventArgs) Handles m_CreateEngineIfAllModesCheckBox.Click
1872         Me.m_CreateEngineIfAllModesCheckBox.Checked = Not (Me.
     m_CreateEngineIfAllModesCheckBox.CheckState = CheckState.Indeterminate)
1873
1874         If Me.m_CreateEngineIfAllModesCheckBox.CheckState = CheckState.Checked Then
1875             Me.SetCreateEngineModes(True)
1876         ElseIf Me.m_CreateEngineIfAllModesCheckBox.CheckState = CheckState.Unchecked Then
1877             Me.SetCreateEngineModes(False)
1878         End If
1879     End Sub
1880
1881     Private Sub OnCreateTransmissionIfAllModesCheckBox_Click(ByVal sender As System.Object
     , ByVal e As System.EventArgs) Handles m_CreateTransmissionIfAllModesCheckBox.Click
1882         Me.m_CreateTransmissionIfAllModesCheckBox.Checked = Not (Me.
     m_CreateTransmissionIfAllModesCheckBox.CheckState = CheckState.Indeterminate)
1883
1884         If Me.m_CreateTransmissionIfAllModesCheckBox.CheckState = CheckState.Checked Then
1885             Me.SetCreateTransmissionModes(True)
1886         ElseIf Me.m_CreateTransmissionIfAllModesCheckBox.CheckState = CheckState.Unchecked
      Then
1887             Me.SetCreateTransmissionModes(False)
1888         End If
1889     End Sub
1890
1891     Private Sub OnSkipEngineTransmissionIfAllModesCheckBox_Click(ByVal sender As System.
     Object, ByVal e As System.EventArgs) Handles
     m_SkipEngineTransmissionIfAllModesCheckBox.Click
1892         Me.m_SkipEngineTransmissionIfAllModesCheckBox.Checked = Not (Me.
     m_SkipEngineTransmissionIfAllModesCheckBox.CheckState = CheckState.Indeterminate)
1893
1894         If Me.m_SkipEngineTransmissionIfAllModesCheckBox.CheckState = CheckState.Checked
     Then
1895             Me.SetSkipEngineTransmissionModes(True)
1896         ElseIf Me.m_SkipEngineTransmissionIfAllModesCheckBox.CheckState = CheckState.
     Unchecked Then
1897             Me.SetSkipEngineTransmissionModes(False)
1898         End If
1899     End Sub
1900
1901     Private Sub SetOperatingModes(ByVal Checked As Boolean)
1902         Me.m_NoFinesModeCheckBox.Checked = Checked
1903         Me.m_LowCostTechnologyFirstModeCheckBox.Checked = Checked
1904         Me.m_AutoInvokeEfficiencyModeCheckBox.Checked = Checked
1905         Me.m_BackFillTechnologyModeCheckBox.Checked = Checked
1906         Me.m_StageAnalysisCheckBox.Checked = Checked
1907         Me.m_MonteCarloAnalysisCheckBox.Checked = Checked
1908         Me.SetCreateEngineModes(Checked)
1909         Me.SetCreateTransmissionModes(Checked)
1910         Me.SetSkipEngineTransmissionModes(Checked)
1911         Me.m_ReserveTechsIfInComplianceModeCheckBox.Checked = Checked
1912
1913         Me.ChangeDetected = True
```

```
1914    End Sub
1915
1916    Private Function GetOperatingModesCheckedState() As Integer
1917        If Me.m_NoFinesModeCheckBox.Checked AndAlso _
1918           Me.m_LowCostTechnologyFirstModeCheckBox.Checked AndAlso _
1919           Me.m_AutoInvokeEfficiencyModeCheckBox.Checked AndAlso _
1920           Me.m_BackFillTechnologyModeCheckBox.Checked AndAlso _
1921           Me.m_StageAnalysisCheckBox.Checked AndAlso _
1922           Me.m_MonteCarloAnalysisCheckBox.Checked AndAlso _
1923           Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Checked AndAlso _
1924           Me.m_CreateEngineIfMultipleRCModeCheckBox.Checked AndAlso _
1925           Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Checked AndAlso _
1926           Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Checked AndAlso _
1927           Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Checked AndAlso _
1928           Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Checked AndAlso _
1929           Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Checked AndAlso _
1930           Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Checked AndAlso _
1931           Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Checked AndAlso _
1932           Me.m_ReserveTechsIfInComplianceModeCheckBox.Checked Then
1933            '
1934            Return 1
1935        ElseIf Me.m_NoFinesModeCheckBox.Checked OrElse _
1936               Me.m_LowCostTechnologyFirstModeCheckBox.Checked OrElse _
1937               Me.m_AutoInvokeEfficiencyModeCheckBox.Checked OrElse _
1938               Me.m_BackFillTechnologyModeCheckBox.Checked OrElse _
1939               Me.m_StageAnalysisCheckBox.Checked OrElse _
1940               Me.m_MonteCarloAnalysisCheckBox.Checked OrElse _
1941               Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Checked OrElse _
1942               Me.m_CreateEngineIfMultipleRCModeCheckBox.Checked OrElse _
1943               Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Checked OrElse _
1944               Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Checked OrElse _
1945               Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Checked OrElse _
1946               Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Checked OrElse _
1947               Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Checked OrElse _
1948               Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Checked OrElse _
1949               Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Checked OrElse ↙
     _
1950               Me.m_ReserveTechsIfInComplianceModeCheckBox.Checked Then
1951            '
1952            Return 2
1953        Else
1954            Return 0
1955        End If
1956    End Function
1957
1958    Private Sub SetCreateEngineModes(ByVal Checked As Boolean)
1959        Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Checked = Checked
1960        Me.m_CreateEngineIfMultipleRCModeCheckBox.Checked = Checked
1961        Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Checked = Checked
1962
1963        Me.ChangeDetected = True
1964    End Sub
1965
1966    Private Function GetCreateEngineModesCheckedState() As Integer
1967        If Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Checked AndAlso _
1968           Me.m_CreateEngineIfMultipleRCModeCheckBox.Checked AndAlso _
1969           Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Checked Then
1970            '
1971            Return 1
1972        ElseIf Me.m_CreateEngineIfCarsInvolvedModeCheckBox.Checked OrElse _
1973               Me.m_CreateEngineIfMultipleRCModeCheckBox.Checked OrElse _
1974               Me.m_CreateEngineIfTechEnableMissMatchModeCheckBox.Checked Then
1975            '
1976            Return 2
1977        Else
1978            Return 0
1979        End If
1980    End Function
1981
1982    Private Sub SetCreateTransmissionModes(ByVal Checked As Boolean)
```

```
1983        Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Checked = Checked
1984        Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Checked = Checked
1985        Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Checked = Checked
1986
1987        Me.ChangeDetected = True
1988    End Sub
1989
1990    Private Function GetCreateTransmissionModesCheckedState() As Integer
1991        If Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Checked AndAlso _
1992           Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Checked AndAlso _
1993           Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Checked Then
1994            '
1995            Return 1
1996        ElseIf Me.m_CreateTransmissionIfCarsInvolvedModeCheckBox.Checked OrElse _
1997               Me.m_CreateTransmissionIfMultipleRCModeCheckBox.Checked OrElse _
1998               Me.m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.Checked Then
1999            '
2000            Return 2
2001        Else
2002            Return 0
2003        End If
2004    End Function
2005
2006    Private Sub SetSkipEngineTransmissionModes(ByVal Checked As Boolean)
2007        Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Checked = Checked
2008        Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Checked = Checked
2009        Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Checked = Checked
2010
2011        Me.ChangeDetected = True
2012    End Sub
2013
2014    Private Function GetSkipEngineTransmissionModesCheckedState() As Integer
2015        If Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Checked AndAlso _
2016           Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Checked AndAlso _
2017           Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Checked Then
2018            '
2019            Return 1
2020        ElseIf Me.m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.Checked OrElse _
2021               Me.m_SkipEngineTransmissionIfMultipleRCModeCheckBox.Checked OrElse _
2022               Me.m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.Checked Then
2023            '
2024            Return 2
2025        Else
2026            Return 0
2027        End If
2028    End Function
2029
2030    Private Sub OnNoFinesModeCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal ↙
        e As System.EventArgs) Handles m_NoFinesModeCheckBox.CheckedChanged
2031        Me.ChangeDetected = True
2032    End Sub
2033
2034    Private Sub OnLowCostTechnologyFirstModeCheckBox_CheckedChanged(ByVal sender As System ↙
        .Object, ByVal e As System.EventArgs) Handles m_LowCostTechnologyFirstModeCheckBox. ↙
        CheckedChanged
2035        Me.ChangeDetected = True
2036    End Sub
2037
2038    Private Sub OnAutoInvokeEfficiencyModeCheckBox_CheckedChanged(ByVal sender As System. ↙
        Object, ByVal e As System.EventArgs) Handles m_AutoInvokeEfficiencyModeCheckBox. ↙
        CheckedChanged
2039        Me.ChangeDetected = True
2040    End Sub
2041
2042    Private Sub OnBackFillTechnologyModeCheckBox_CheckedChanged(ByVal sender As System. ↙
        Object, ByVal e As System.EventArgs) Handles m_BackFillTechnologyModeCheckBox. ↙
        CheckedChanged
2043        Me.ChangeDetected = True
2044    End Sub
2045
```

```
2046    Private Sub OnStageAnalysisCheckBox_CheckedChanged(ByVal sender As System.Object,
        ByVal e As System.EventArgs) Handles m_StageAnalysisCheckBox.CheckedChanged
2047        Me.ChangeDetected = True
2048    End Sub
2049
2050    Private Sub OnMonteCarloAnalysisCheckBox_CheckedChanged(ByVal sender As System.Object,
         ByVal e As System.EventArgs) Handles m_MonteCarloAnalysisCheckBox.CheckedChanged
2051        Me.ChangeDetected = True
2052    End Sub
2053
2054    Private Sub OnMonteCarloIterationsNumericUpDown_ValueChanged(ByVal sender As System.
        Object, ByVal e As System.EventArgs) Handles m_MonteCarloIterationsNumericUpDown.
        ValueChanged, m_MonteCarloIterationsNumericUpDown.TextChanged
2055        If Me.m_MonteCarloIterationsNumericUpDown.Value > Me.
        m_MonteCarloIterationsNumericUpDown.Maximum Then
2056            Me.m_MonteCarloIterationsNumericUpDown.Value = Me.
        m_MonteCarloIterationsNumericUpDown.Maximum
2057        End If
2058        Me.ChangeDetected = True
2059    End Sub
2060
2061    Private Sub OnCreateEngineIfCarsInvolvedModeCheckBox_CheckedChanged(ByVal sender As
        System.Object, ByVal e As System.EventArgs) Handles
        m_CreateEngineIfCarsInvolvedModeCheckBox.CheckedChanged
2062        Me.ChangeDetected = True
2063    End Sub
2064
2065    Private Sub OnCreateEngineIfMultipleRCModeCheckBox_CheckedChanged(ByVal sender As
        System.Object, ByVal e As System.EventArgs) Handles
        m_CreateEngineIfMultipleRCModeCheckBox.CheckedChanged
2066        Me.ChangeDetected = True
2067    End Sub
2068
2069    Private Sub OnCreateEngineIfTechEnableMissMatchModeCheckBox_CheckedChanged(ByVal
        sender As System.Object, ByVal e As System.EventArgs) Handles
        m_CreateEngineIfTechEnableMissMatchModeCheckBox.CheckedChanged
2070        Me.ChangeDetected = True
2071    End Sub
2072
2073    Private Sub OnCreateTransmissionIfCarsInvolvedModeCheckBox_CheckedChanged(ByVal sender
         As System.Object, ByVal e As System.EventArgs) Handles
        m_CreateTransmissionIfCarsInvolvedModeCheckBox.CheckedChanged
2074        Me.ChangeDetected = True
2075    End Sub
2076
2077    Private Sub OnCreateTransmissionIfMultipleRCModeCheckBox_CheckedChanged(ByVal sender
        As System.Object, ByVal e As System.EventArgs) Handles
        m_CreateTransmissionIfMultipleRCModeCheckBox.CheckedChanged
2078        Me.ChangeDetected = True
2079    End Sub
2080
2081    Private Sub OnCreateTransmissionIfTechEnableMissMatchModeCheckBox_CheckedChanged(ByVal
         sender As System.Object, ByVal e As System.EventArgs) Handles
        m_CreateTransmissionIfTechEnableMissMatchModeCheckBox.CheckedChanged
2082        Me.ChangeDetected = True
2083    End Sub
2084
2085    Private Sub OnSkipEngineTransmissionIfCarsInvolvedModeCheckBox_CheckedChanged(ByVal
        sender As System.Object, ByVal e As System.EventArgs) Handles
        m_SkipEngineTransmissionIfCarsInvolvedModeCheckBox.CheckedChanged
2086        Me.ChangeDetected = True
2087    End Sub
2088
2089    Private Sub OnSkipEngineTransmissionIfMultipleRCModeCheckBox_CheckedChanged(ByVal
        sender As System.Object, ByVal e As System.EventArgs) Handles
        m_SkipEngineTransmissionIfMultipleRCModeCheckBox.CheckedChanged
2090        Me.ChangeDetected = True
2091    End Sub
2092
2093    Private Sub OnSkipEngineTransmissionIfTechEnableMissMatchModeCheckBox_CheckedChanged
```

```
    (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    m_SkipEngineTransmissionIfTechEnableMissMatchModeCheckBox.CheckedChanged
2094        Me.ChangeDetected = True
2095    End Sub
2096
2097    Private Sub OnReserveTechsIfInComplianceModeCheckBox_CheckedChanged(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
    m_ReserveTechsIfInComplianceModeCheckBox.CheckedChanged
2098        Me.ChangeDetected = True
2099    End Sub
2100
2101 #End Region
2102
2103
2104 #Region " Encryption Settings "
2105
2106    ' File Saving Settings
2107    '
2108    Private Sub OnEncryptOutputFilesCheckBox_CheckedChanged(ByVal sender As System.Object,
     ByVal e As System.EventArgs) Handles m_EncryptOutputFilesCheckBox.CheckedChanged
2109        Me.m_EncryptOutputFilesPanel.Enabled = Me.m_EncryptOutputFilesCheckBox.Checked
2110        Me.ChangeDetected = True
2111    End Sub
2112
2113    Private Sub OnEncryptionProviderComboBox_SelectedIndexChanged(ByVal sender As System.
    Object, ByVal e As System.EventArgs) Handles m_EncryptionProviderComboBox.
    SelectedIndexChanged
2114        Me.ChangeDetected = True
2115    End Sub
2116
2117    Private Sub OnKeyLengthNumericUpDown_ValueChanged(ByVal sender As System.Object, ByVal
     e As System.EventArgs) Handles m_KeyLengthNumericUpDown.ValueChanged
2118        Me.ChangeDetected = True
2119    End Sub
2120
2121    Private Sub OnEncryptDocPropertiesCheckBox_CheckedChanged(ByVal sender As System.
    Object, ByVal e As System.EventArgs) Handles m_EncryptDocPropertiesCheckBox.
    CheckedChanged
2122        Me.ChangeDetected = True
2123    End Sub
2124
2125    Private Sub OnPasswordToModifyCheckBox_CheckedChanged(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles m_PasswordToModifyCheckBox.CheckedChanged
2126        If Me.m_PasswordToModifyCheckBox.Checked Then
2127            Me.m_PasswordToModifyTextBox.Text = Me.m_PasswordToOpenTextBox.Text
2128        Else
2129            Me.m_PasswordToModifyTextBox.Text = ""
2130        End If
2131
2132        Me.m_PasswordToModifyTextBox.ReadOnly = Me.m_PasswordToModifyCheckBox.Checked
2133        Me.ChangeDetected = True
2134    End Sub
2135
2136    Private Sub OnPasswordToOpenTextBox_TextChanged(ByVal sender As System.Object, ByVal e
     As System.EventArgs) Handles m_PasswordToOpenTextBox.TextChanged
2137        If Me.m_PasswordToModifyCheckBox.Checked Then
2138            Me.m_PasswordToModifyCheckBox.Checked = False
2139        End If
2140        '
2141        Me.ChangeDetected = True
2142    End Sub
2143
2144    Private Sub OnPasswordToModifyTextBox_TextChanged(ByVal sender As System.Object, ByVal
     e As System.EventArgs) Handles m_PasswordToModifyTextBox.TextChanged
2145        Me.ChangeDetected = True
2146    End Sub
2147
2148    ' File Loading Settings
2149    '
2150    Private Sub OnReadEncryptedFilesCheckBox_CheckedChanged(ByVal sender As System.Object,
```

```
        ByVal e As System.EventArgs) Handles m_ReadEncryptedFilesCheckBox.CheckedChanged
2151        Me.m_ReadEncryptedFilesPanel.Enabled = Me.m_ReadEncryptedFilesCheckBox.Checked
2152        Me.ChangeDetected = True
2153        If Me.m_ReadEncryptedFilesCheckBox.Checked Then
2154            Me.m_InputSinglePwTextBox.SelectAll()
2155            Me.m_InputSinglePwTextBox.Focus()
2156        End If
2157    End Sub
2158
2159    Private Sub OnInputSinglePwCheckBox_CheckedChanged(ByVal sender As System.Object,
        ByVal e As System.EventArgs) Handles m_InputSinglePwCheckBox.CheckedChanged
2160        ' enable the textbox for a single pw
2161        Me.m_InputSinglePwTextBox.Enabled = Me.m_InputSinglePwCheckBox.Checked
2162        ' uncheck the multi pw checkbox and disable the multi pw textboxes
2163        If Me.m_InputSinglePwCheckBox.Checked Then
2164            Me.m_InputMultiPwCheckBox.Checked = False
2165            Me.m_InputMultiPwVehLabel.Enabled = False
2166            Me.m_InputMultiPwVehTextBox.Enabled = False
2167            Me.m_InputMultiPwEngLabel.Enabled = False
2168            Me.m_InputMultiPwEngTextBox.Enabled = False
2169            Me.m_InputMultiPwTranLabel.Enabled = False
2170            Me.m_InputMultiPwTranTextBox.Enabled = False
2171            Me.m_InputMultiPwTechLabel.Enabled = False
2172            Me.m_InputMultiPwTechTextBox.Enabled = False
2173        End If
2174        '
2175        Me.ChangeDetected = True
2176    End Sub
2177
2178    Private Sub OnInputMultiPwCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal
         e As System.EventArgs) Handles m_InputMultiPwCheckBox.CheckedChanged
2179        ' enable the textbox for multi pw
2180        Me.m_InputMultiPwVehLabel.Enabled = Me.m_InputMultiPwCheckBox.Checked
2181        Me.m_InputMultiPwVehTextBox.Enabled = Me.m_InputMultiPwCheckBox.Checked
2182        Me.m_InputMultiPwEngLabel.Enabled = Me.m_InputMultiPwCheckBox.Checked
2183        Me.m_InputMultiPwEngTextBox.Enabled = Me.m_InputMultiPwCheckBox.Checked
2184        Me.m_InputMultiPwTranLabel.Enabled = Me.m_InputMultiPwCheckBox.Checked
2185        Me.m_InputMultiPwTranTextBox.Enabled = Me.m_InputMultiPwCheckBox.Checked
2186        Me.m_InputMultiPwTechLabel.Enabled = Me.m_InputMultiPwCheckBox.Checked
2187        Me.m_InputMultiPwTechTextBox.Enabled = Me.m_InputMultiPwCheckBox.Checked
2188        ' uncheck the single pw checkbox and disable the single pw textbox
2189        If Me.m_InputMultiPwCheckBox.Checked Then
2190            Me.m_InputSinglePwCheckBox.Checked = False
2191            Me.m_InputSinglePwTextBox.Enabled = False
2192        End If
2193        '
2194        Me.ChangeDetected = True
2195    End Sub
2196
2197    Private Sub OnInputSinglePwTextBox_TextChanged(ByVal sender As System.Object, ByVal e
        As System.EventArgs) Handles m_InputSinglePwTextBox.TextChanged
2198        Me.ChangeDetected = True
2199    End Sub
2200
2201    Private Sub OnInputMultiPwVehTextBox_TextChanged(ByVal sender As System.Object, ByVal
        e As System.EventArgs) Handles m_InputMultiPwVehTextBox.TextChanged
2202        Me.ChangeDetected = True
2203    End Sub
2204
2205    Private Sub OnInputMultiPwEngTextBox_TextChanged(ByVal sender As System.Object, ByVal
        e As System.EventArgs) Handles m_InputMultiPwEngTextBox.TextChanged
2206        Me.ChangeDetected = True
2207    End Sub
2208
2209    Private Sub OnInputMultiPwTranTextBox_TextChanged(ByVal sender As System.Object, ByVal
         e As System.EventArgs) Handles m_InputMultiPwTranTextBox.TextChanged
2210        Me.ChangeDetected = True
2211    End Sub
2212
2213    Private Sub OnInputMultiPwTechTextBox_TextChanged(ByVal sender As System.Object, ByVal
```

```
        e As System.EventArgs) Handles m_InputMultiPwTechTextBox.TextChanged
2214        Me.ChangeDetected = True
2215    End Sub
2216
2217 #End Region
2218
2219
2220 #Region " Output Settings "
2221
2222    ' General
2223    '
2224    Private Sub OnDefaultOutputPathTextBox_TextChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles m_DefaultOutputPathTextBox.TextChanged
2225        Me.ChangeDetected = True
2226    End Sub
2227
2228    Private Sub OnDefaultOutputPathBrowseButton(ByVal sender As System.Object, ByVal e As
       System.EventArgs) Handles m_DefaultOutputPathBrowseButton.Click
2229        If Me.FolderBrowserDialogHelper(Me.m_DefaultOutputPathTextBox, "Please select the
       default output path.") Then
2230            Me.ChangeDetected = True
2231        End If
2232    End Sub
2233
2234    Private Sub OnHideUnusedRCsCheckBox_CheckedChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles m_HideUnusedRCsCheckBox.CheckedChanged
2235        Me.ChangeDetected = True
2236    End Sub
2237
2238    ' Output Templates
2239    '
2240    Private Sub OnUseOutputTemplatesCheckBox_CheckedChanged(ByVal sender As System.Object,
        ByVal e As System.EventArgs) Handles m_UseOutputTemplatesCheckBox.CheckedChanged
2241        Me.m_UseOutputTemplatesCheckBox2.Checked = Me.m_UseOutputTemplatesCheckBox.Checked
2242        Me.ChangeDetected = True
2243    End Sub
2244
2245    Private Sub OnConfigureTemplatesButton(ByVal sender As System.Object, ByVal e As
       System.EventArgs) Handles m_ConfigureTemplatesButton.Click
2246        'Me.m_OptionsTabControl.SelectedTab = Me.m_TemplatesTabPage
2247    End Sub
2248
2249    ' Report Settings
2250    '
2251    Private Sub OnGenAllReportsCheckBox_Click(ByVal sender As System.Object, ByVal e As
       System.EventArgs) Handles m_GenAllReportsCheckBox.Click
2252        Me.m_GenAllReportsCheckBox.Checked = Not (Me.m_GenAllReportsCheckBox.CheckState =
       CheckState.Indeterminate)
2253
2254        If Me.m_GenAllReportsCheckBox.CheckState = CheckState.Checked Then
2255            Me.m_GenIndustryReportCheckBox.Checked = True
2256            Me.m_GenMfgsReportCheckBox.Checked = True
2257            Me.m_GenVehiclesReportCheckBox.Checked = True
2258            Me.m_GenEffectsReportCheckBox.Checked = True
2259            Me.m_GenEAReportCheckBox.Checked = True
2260        ElseIf Me.m_GenAllReportsCheckBox.CheckState = CheckState.Unchecked Then
2261            Me.m_GenIndustryReportCheckBox.Checked = False
2262            Me.m_GenMfgsReportCheckBox.Checked = False
2263            Me.m_GenVehiclesReportCheckBox.Checked = False
2264            Me.m_GenEffectsReportCheckBox.Checked = False
2265            Me.m_GenEAReportCheckBox.Checked = False
2266        End If
2267
2268        Me.ChangeDetected = True
2269    End Sub
2270
2271    Private Sub OnGenIndustryReportCheckBox_CheckedChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles m_GenIndustryReportCheckBox.CheckedChanged
2272        Me.ChangeDetected = True
2273    End Sub
```

```
2274
2275        Private Sub OnGenMfgReportCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal
             e As System.EventArgs) Handles m_GenMfgsReportCheckBox.CheckedChanged
2276            Me.ChangeDetected = True
2277        End Sub
2278
2279        Private Sub OnGenVehicleReportCheckBox_CheckedChanged(ByVal sender As System.Object,
            ByVal e As System.EventArgs) Handles m_GenVehiclesReportCheckBox.CheckedChanged
2280            Me.ChangeDetected = True
2281        End Sub
2282
2283        Private Sub OnGenEffectsReportCheckBox_CheckedChanged(ByVal sender As System.Object,
            ByVal e As System.EventArgs) Handles m_GenEffectsReportCheckBox.CheckedChanged
2284            Me.ChangeDetected = True
2285        End Sub
2286
2287        Private Sub OnGenEAReportCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal
            e As System.EventArgs) Handles m_GenEAReportCheckBox.CheckedChanged
2288            Me.ChangeDetected = True
2289        End Sub
2290
2291    #End Region
2292
2293
2294    #Region " Template Settings "
2295
2296        ' Output Templates
2297        '
2298        Private Sub OnUseOutputTemplatesCheckBox2_CheckedChanged(ByVal sender As System.Object
            , ByVal e As System.EventArgs) Handles m_UseOutputTemplatesCheckBox2.CheckedChanged
2299            Me.m_UseOutputTemplatesCheckBox.Checked = Me.m_UseOutputTemplatesCheckBox2.Checked
2300            Me.ChangeDetected = True
2301        End Sub
2302
2303        Private Sub OnDefaultOutputTemplateBrowseButton(ByVal sender As System.Object, ByVal e
             As System.EventArgs) Handles m_DefaultOutputTemplateBrowseButton2.Click,
            m_DefaultOutputTemplateBrowseButton.Click
2304            Dim OrigPath As String = Me.m_DefaultOutputTemplateTextBox.Text
2305
2306            If Me.FileBrowserDialogHelper(Me.m_DefaultOutputTemplateTextBox, _
2307                                          "Please select the default output template.") Then
2308                If CheckOutputTemplatePaths(Me.m_DefaultOutputTemplateTextBox, OrigPath) Then
2309                    Me.m_DefaultOutputTemplateTextBox2.Text = m_DefaultOutputTemplateTextBox.
        Text
2310                    Me.ChangeDetected = True
2311                End If
2312            End If
2313        End Sub
2314
2315        Private Sub OnLandscapeOutputTemplateBrowseButton(ByVal sender As System.Object, ByVal
             e As System.EventArgs) Handles m_LandscapeOutputTemplateBrowseButton.Click
2316            Dim OrigPath As String = Me.m_LandscapeOutputTemplateTextBox.Text
2317
2318            If Me.FileBrowserDialogHelper(Me.m_LandscapeOutputTemplateTextBox, _
2319                                          "Please select an output template for generating " &
         _
2320                                          "reports with landscape orientation.") Then
2321                '
2322                If CheckOutputTemplatePaths(Me.m_LandscapeOutputTemplateTextBox, OrigPath)
        Then
2323                    Me.ChangeDetected = True
2324                End If
2325            End If
2326        End Sub
2327
2328        Private Sub OnPortraitOutputTemplateBrowseButton(ByVal sender As System.Object, ByVal
            e As System.EventArgs) Handles m_PortraitOutputTemplateBrowseButton.Click
2329            Dim OrigPath As String = Me.m_PortraitOutputTemplateTextBox.Text
2330
2331            If Me.FileBrowserDialogHelper(Me.m_PortraitOutputTemplateTextBox, _
```

```
2332                                        "Please select an output template for generating " & _
2333                                        "reports with portrait orientation.") Then
2334                '
2335                If CheckOutputTemplatePaths(Me.m_PortraitOutputTemplateTextBox, OrigPath) Then
2336                    Me.ChangeDetected = True
2337                End If
2338            End If
2339        End Sub
2340
2341        Private Function CheckOutputTemplatePaths(ByRef Control As System.Windows.Forms.TextBox, _
2342                                                  ByVal OrigPath As String) As Boolean
2343            '
2344            Return True
2345
2346            '
2347            ' Currently, there is only one template available, so the template checks are not required
2348            '
2349            '''If Not CheckOutputTemplatePath(Control, Me.m_DefaultOutputTemplateTextBox) OrElse _
2350            '''   Not CheckOutputTemplatePath(Control, Me.m_LandscapeOutputTemplateTextBox) OrElse _
2351            '''   Not CheckOutputTemplatePath(Control, Me.m_PortraitOutputTemplateTextBox) Then
2352            '''    '
2353            '''    MsgBox("This template has already been selected for use elsewhere.  " & _
2354            '''           "Please select a different template", MsgBoxStyle.Information, _
2355            '''           "Template already in use.")
2356            '''    Control.Text = OrigPath
2357            '''    Return False
2358            '''Else
2359            '''    Return True
2360            '''End If
2361        End Function
2362
2363        Private Function CheckOutputTemplatePath(ByRef Control1 As System.Windows.Forms.TextBox, _
2364                                                 ByRef Control2 As System.Windows.Forms.TextBox) As Boolean
2365            '
2366            Return (Not (Control1 Is Control2)) AndAlso (Control1.Text = Control2.Text)
2367        End Function
2368
2369    #End Region
2370
2371
2372    #Region " Other Settings "
2373
2374        Private Sub OnRunEffectsModelCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles m_RunEffectsModelCheckBox.CheckedChanged
2375            Me.ChangeDetected = True
2376        End Sub
2377
2378        Private Sub OnNoStartupWarningCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles m_NoStartupWarningCheckBox.CheckedChanged
2379            Me.ChangeDetected = True
2380        End Sub
2381
2382        Private Sub OnSingleFileInputCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles m_SingleFileInputCheckBox.Click
2383            Me.ChangeDetected = True
2384            Me.m_SingleFileInputCheckBox.Checked = Me.m_SingleFileInputCheckBox.Checked
2385            Me.m_ThreeFileInputCheckBox.Checked = Not Me.m_SingleFileInputCheckBox.Checked
2386        End Sub
2387
2388        Private Sub OnThreeFileInputCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles m_ThreeFileInputCheckBox.Click
2389            Me.ChangeDetected = True
```

```
2390            Me.m_SingleFileInputCheckBox.Checked = Not Me.m_ThreeFileInputCheckBox.Checked
2391            Me.m_ThreeFileInputCheckBox.Checked = Me.m_ThreeFileInputCheckBox.Checked
2392        End Sub
2393
2394        ' model profiling
2395        Private Sub m_EnableProfilingCheckBox_CheckedChanged(ByVal sender As System.Object,
            ByVal e As System.EventArgs) Handles m_EnableProfilingCheckBox.CheckedChanged
2396            Me.ChangeDetected = True
2397        End Sub
2398
2399        Private Sub m_EnableProfilingMemoryCheckBox_CheckedChanged(ByVal sender As System.
            Object, ByVal e As System.EventArgs) Handles m_EnableProfilingMemoryCheckBox.
            CheckedChanged
2400            Me.ChangeDetected = True
2401        End Sub
2402
2403    #End Region
2404
2405
2406        Private Sub m_TechSettingsTabPage_Click(ByVal sender As System.Object, ByVal e As
            System.EventArgs) Handles m_TechSettingsTabPage.Click
2407
2408        End Sub
2409    End Class
2410
```

```
Imports System
Imports System.IO

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
'''
Public Class Output

    Private Shared Reports As OutputReports     ' stores the active reports reference
    Private Shared IsActive As Boolean          ' specifies if the generating of reports is active (running)
    Private Shared Status As String = "Generating reports:  "

    ' track the report errors
    Private Shared nErrors As Integer
    Private Shared Errors() As String
    Private Shared NoErrors As Boolean

    Private Shared ShowStatistics As Boolean = False

    ' these variables store the run times of each reports (time required to produce each report)
    Private Shared _START_TIME_ As Long         ' start the timer
    Private Shared _END_TIME_ As Long           ' end the timer
    Public Shared SummaryRunTime() As Long
    Public Shared IndustryRunTime() As Long
    Public Shared ManufacturerRunTime() As Long
    Public Shared VehiclesRunTime() As Long
    Public Shared EffectsRunTime() As Long
    Public Shared EAsRunTimes As Long

    '''
    ' This mehod produces formated reports.
    '
    Public Shared Function GenerateReports(ByVal GetPath As Boolean, _
                                           Optional ByRef nReportErrors As Integer = 0, _
                                           Optional ByRef ReportErrors() As String = Nothing) As Boolean
        '
        Dim i As Integer
        Dim RunReports As Boolean = True
        Dim nScenarios As Integer
        Dim path As String

        If Not IsNothing(CAFE.Profiling) Then
            CAFE.Profiling.OpenLast()
        End If

        ' there are no report errors initially
        Output.NoErrors = True
        Output.nErrors = nReportErrors
        Output.Errors = ReportErrors

        Output.NoErrors = PerformInitialChecks()

        If Output.NoErrors Then
            If GetPath Then
                path = GetOutputPath()    ' get output path
            Else
                path = CAFE.OutputSettings.DefaultOutputPath
            End If

            If IsNothing(path) OrElse path.Trim() = "" Then
                ReDim Preserve Errors(nErrors)
                Errors(nErrors) = "No path selected..."
                Debug.WriteLine(Errors(nErrors))
                nErrors += 1
                Output.NoErrors = False
```

```
66              End If
67          End If
68
69          If Output.NoErrors Then
70              nScenarios = CAFE.Scenarios.nScenarios
71
72              ReDim SummaryRunTime(nScenarios - 1)
73              ReDim IndustryRunTime(nScenarios - 1)
74              ReDim ManufacturerRunTime(nScenarios - 1)
75              ReDim VehiclesRunTime(nScenarios - 1)
76              ReDim EffectsRunTime(nScenarios - 1)
77
78              With CAFE.OutputSettings
79                  If Not .GenerateIndustryReport AndAlso Not .GenerateMfgsReport AndAlso _
80                     Not .GenerateVehiclesReport AndAlso Not .GenerateEffectsReport AndAlso _
81                     Not .GenerateEnvironmentalAssessmentReport Then
82                      '
83                      ReDim Preserve Errors(nErrors)
84                      Errors(nErrors) = "No reports selected for output.  Please visit the options " & _
85                                        "window to select the reports to be generated, and try again."
86                      MsgBox(Errors(nErrors), MsgBoxStyle.Information, "No Reports Selected")
87                      nErrors += 1
88                      Output.NoErrors = False
89                  End If
90              End With
91          End If
92
93          If Output.NoErrors Then
94              CAFE.CAFE_Form.SetStatusBarStatus(Status + "preparing to generate reports...")
95              Output.IsActive = True
96              Application.DoEvents()
97
98              Output.Reports = New OutputReports
99
100             For i = 0 To nScenarios - 1
101                 CAFE.CAFE_Form.SetStatusBarScenario(i)
102
103                 Reports.Scenario = i
104                 Reports.path = path & "\Scenario_" & i
105
106                 ' first, write summary sheet
107                 Output.GenerateReport(OutputReports.ReportTypeEnum.SUMMARY, "summary", Output.SummaryRunTime(i))
108
109                 ' write industry report
110                 If CAFE.OutputSettings.GenerateIndustryReport Then
111                     Output.GenerateReport(OutputReports.ReportTypeEnum.INDUSTRY, "industry", Output.IndustryRunTime(i))
112                 End If
113
114                 ' write manufacturer report
115                 If CAFE.OutputSettings.GenerateMfgsReport Then
116                     Output.GenerateReport(OutputReports.ReportTypeEnum.MANUFACTURER, "manufacturers", Output.ManufacturerRunTime(i))
117                 End If
118
119                 ' write vehicles report
120                 If CAFE.OutputSettings.GenerateVehiclesReport Then
121                     Output.GenerateReport(OutputReports.ReportTypeEnum.VEHICLES, "vehicles", Output.VehiclesRunTime(i))
122                 End If
123
124                 ' write effects report
125                 If CAFE.OtherSettings.RunEffectsModel AndAlso CAFE.OutputSettings.GenerateEffectsReport Then
126                     Output.GenerateReport(OutputReports.ReportTypeEnum.EFFECTS, "effects", Output.EffectsRunTime(i))
```

```
127                 End If
128             Next i
129
130             ' write environmental assessment report
131             If CAFE.OutputSettings.GenerateEnvironmentalAssessmentReport Then
132                 Output.GenerateReport(OutputReports.ReportTypeEnum.EAS, "environmental assessment", Output.EAsRunTimes)
133             End If
134
135             Output.Reports.CloseTemplates()
136             Application.DoEvents()
137
138             Output.DisplayRuntimeStatistics(nScenarios)
139
140             ' check if user canceled reports
141             If Not Output.IsActive Then
142                 ReDim Preserve Output.Errors(Output.nErrors)
143                 Output.Errors(Output.nErrors) = "Generating of reports aborted by user."
144                 Output.nErrors += 1
145                 Output.NoErrors = False
146             End If
147
148             ' the reports are done, set active to false
149             Output.IsActive = False
150         End If
151
152         ' set the errors, if any
153         nReportErrors = Output.nErrors
154         ReportErrors = Output.Errors
155
156         If Not IsNothing(CAFE.Profiling) Then
157             CAFE.Profiling.Close()
158         End If
159
160         Return NoErrors
161     End Function
162
163     '''
164     ' This function performs inital checks to determine if the reports can be run at this time.
165     '
166     Private Shared Function PerformInitialChecks() As Boolean
167         CAFE.CAFE_Form.SetStatusBarStatus(Status + "performing initial checks...")
168
169         If Not CAFE.ModelingComplete Then
170             ' the model hasn't even run yet...exit
171             ReDim Preserve Errors(nErrors + 1)
172             Errors(nErrors) = "Model must be run prior to generating the reports."
173             nErrors += 1
174             Errors(nErrors) = "Run the model first!"
175             nErrors += 1
176             Debug.WriteLine(Errors(nErrors - 1))
177             Return False
178         End If
179
180         If Not CAFE.ModelingSuccessful Then
181             ' the model did not complete successfully
182             ReDim Preserve Errors(nErrors + 1)
183             Errors(nErrors) = "Modeling did not complete successfully; generating of reports aborted."
184             nErrors += 1
185             Errors(nErrors) = "Please run the model again."
186             nErrors += 1
187             Debug.WriteLine(Errors(nErrors - 1))
188             Return False
189         End If
190
191         If IsNothing(CAFE.IndustriesStack) OrElse IsNothing(CAFE.MfgsStack) OrElse IsNothing(CAFE.VehiclesStack) Then
```

```
192             ' the stacks are empty, even though the model has already run
193             ReDim Preserve Errors(nErrors + 1)
194             Errors(nErrors) = "Output stacks are empty--reports cannot be generated.  Model may need to be restarted."
195             nErrors += 1
196             Errors(nErrors) = "Please run the model again."
197             nErrors += 1
198             Debug.WriteLine(Errors(nErrors - 1))
199             Return False
200         End If
201
202         Return True
203     End Function
204
205     '''
206     ' Generates a single report based on the specified parameters.
207     '
208     Private Shared Function GenerateReport(ByVal ReportType As OutputReports.ReportTypeEnum, _
209                                            ByVal ReportStr As String, _
210                                            ByRef ReportRunTime As Long) As Boolean
211         '
212         Application.DoEvents()
213
214         ' check if user aborted generating of reports
215         If Not Output.IsActive Then
216             ' user canceled reports
217             ReDim Preserve Output.Errors(Output.nErrors)
218             Output.Errors(Output.nErrors) = "The report for " & ReportStr & " was not generated " & _
219                                             "because it was aborted by user."
220             Output.nErrors += 1
221
222             Return False
223         End If
224
225         Output.Reports.ReportType = ReportType
226         CAFE.CAFE_Form.SetStatusBarStatus(Status & "writing " & ReportStr & "...")
227         _START_TIME_ = Now.Ticks
228         Try
229             Select Case ReportType
230                 Case OutputReports.ReportTypeEnum.SUMMARY
231                     Output.Reports.WriteSummaryPage()
232                 Case OutputReports.ReportTypeEnum.INDUSTRY
233                     Output.Reports.ReportIndustry()
234                 Case OutputReports.ReportTypeEnum.MANUFACTURER
235                     Output.Reports.ReportManufacturers()
236                 Case OutputReports.ReportTypeEnum.VEHICLES
237                     Output.Reports.ReportVehicles()
238                 Case OutputReports.ReportTypeEnum.EFFECTS
239                     Output.Reports.ReportEffects()
240                 Case OutputReports.ReportTypeEnum.EAS
241                     Output.Reports.ReportEAs()
242                 Case Else
243                     Return False
244             End Select
245         Catch ex As Exception
246             Try
247                 Output.Reports.PrintBuffer(ReportStr)
248             Catch ex2 As Exception
249             End Try
250             Output.Reports.FinalizeReport()
251             If DBUG Or Not DBUG Then
252                 MsgBox("An exception was caught in GenerateReports() when trying to write " & ReportStr & ": " & _
253                        NewLine & NewLine & "    Message: " & ex.Message & _
254                        NewLine & NewLine & "    Source: " & ex.Source & _
255                        NewLine & NewLine & "    Stack Trace: " & ex.StackTrace, _
256                        MsgBoxStyle.Critical, "Exception")
```

```
257
258                 ReDim Preserve Output.Errors(Output.nErrors)
259                 Output.Errors(Output.nErrors) = "An exception was caught in GenerateReports
    () when trying " & _
260                                 "to write " & ReportStr & ": " & ex.Message
261                 Output.nErrors += 1
262                 Output.NoErrors = False
263
264                 Debug.WriteLine(ex.StackTrace)
265             End If
266         End Try
267
268         _END_TIME_ = Now.Ticks
269         ReportRunTime = (_END_TIME_ - _START_TIME_) / 10000          ' in milliseconds
270
271         CAFE.CAFE_Form.SetStatusBarStatus(Status & "writing " & ReportStr & "...done!")
272         Output.Reports.ReportType = OutputReports.ReportTypeEnum.NONE
273         Application.DoEvents()
274
275         Return Output.NoErrors
276     End Function
277
278     '''
279     ' Displays the run time of each report.
280     '
281     Private Shared Sub DisplayRuntimeStatistics(ByVal nScenarios As Integer)
282         Dim i As Integer
283         Dim Statistics As String
284
285         If ShowStatistics Then
286             For i = 0 To nScenarios - 1
287                 Statistics += "  Summary Sn" + i.ToString + ":  " + Output.SummaryRunTime(i)
    .ToString + " ms." + NewLine
288                 Statistics += "  Industry Sn" + i.ToString + ":  " + Output.IndustryRunTime
    (i).ToString + " ms." + NewLine
289                 Statistics += "  Manufacturer Sn" + i.ToString + ":  " + Output.
    ManufacturerRunTime(i).ToString + " ms." + NewLine
290                 Statistics += "  Vehicles Sn" + i.ToString + ":  " + Output.VehiclesRunTime
    (i).ToString + " ms." + NewLine
291                 Statistics += "  Effects Sn" + i.ToString + ":  " + Output.EffectsRunTime(i)
    .ToString + " ms." + NewLine
292                 Statistics += NewLine
293             Next i
294
295             Statistics += "  Environmental Assessment:  " + Output.EAsRunTimes.ToString + "
    ms." + NewLine + NewLine
296
297             MsgBox("Runtime statistics:" & NewLine & NewLine & Statistics, MsgBoxStyle.
    Information, "Statistics")
298         End If
299     End Sub
300
301     '''
302     ' This function cancels the active report generation
303     '
304     Public Shared Function CancelReports()
305         Output.IsActive = False
306     End Function
307
308     '''
309     ' This method retrieves the path where the generated output files should be writen to by
310     ' prompting the user with a FolderBrowserDialog window.
311     '
312     Public Shared Function GetOutputPath() As String
313         Dim path As String
314         Dim dlg As New FolderBrowserDialog
315         dlg.RootFolder = Environment.SpecialFolder.Desktop
316         dlg.SelectedPath = CAFE.OutputSettings.DefaultOutputPath
317         dlg.Description = "Please select an output path."
```

```
        dlg.ShowNewFolderButton = True
        If dlg.ShowDialog() = DialogResult.OK Then
            path = dlg.SelectedPath()
        Else
            path = ""
        End If

        Return path
    End Function

End Class


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
'''
Public Class OutputReports

    ' private structures used for outputs
    Private Structure EffNValsStr
        Public nMfgT As Integer          ' number of manufacturers plus "total"
        Public nFuelT As Integer         ' number of fuel types (gas/diesel) plus "total"
        Public nCrPl As Integer          ' number of criteria pollutants
        Public nM6T As Integer           ' number of mobile 6 classes plus "total"
        Public nSafetyT As Integer       ' number of safety classes plus "total"
    End Structure

    Private Structure EffSectsStr
        Public EmplStart As Integer
        Public ECStart As Integer
        Public EmnsStart As Integer
        Public SftStart As Integer
        Public OaSCostStart As Integer
        Public DOaSCostStart As Integer
    End Structure

    Public Enum ReportTypeEnum
        NONE = -1
        SUMMARY = 0
        INDUSTRY = 1
        MANUFACTURER = 2
        VEHICLES = 3
        EFFECTS = 4
        EAS = 5
    End Enum

    Public ReportType As ReportTypeEnum = ReportTypeEnum.NONE

    ' templates
    Private DefaultTemplate As C_Cafe_InFile
    Private LandscapeTemplate As C_Cafe_InFile
    Private PortraitTemplate As C_Cafe_InFile

    Private range As Excel.Range
    Private file As C_CAFE_OutFile

    Private buffer(,) As Object
    Private row As Integer
    Private col As Integer

    Public path As String
    Public SummaryPath As String      ' the path of the most recent output file
    Public Scenario As Integer
    Public ScenarioName As String
    Public MY As Integer

    Private nRowsToRemove As Integer = 0
    Private RowsToRemove() As Object
```

```
386
387     Private nColsToRemove As Integer = 0
388     Private ColsToRemove() As Object
389
390     '''
391     ' number formats:
392     ' plaint text:          @
393     ' numeric regular:      #,##0
394     ' numeric decimal 1:    #,##0.0
395     ' numeric decimal 2:    #,##0.00
396     ' accounting:           _($* #,##0_);_($* (#,##0);_($* " & """" & " - " & """" & "??_);_ ↙
        (@_)
397     ' accounting decimal:   _($* #,##0.00_);_($* (#,##0.00);_($* " & """" & " - " & """" &  ↙
        "??_);_(@_)
398     '
399     Private NumberFormats() As String = {"#,##0", _
400                                          "#,##0.0", _
401                                          "#,##0.00", _
402                                          "_($* #,##0_);_($* (#,##0);_($* "" - ""??_);_(@_)", ↙
         _
403                                          "_($* #,##0.00_);_($* (#,##0.00);_($* "" - ""??_);_ ↙
        (@_)", _
404                                          "_(* 0%_);_(* (0%);_(* ""- %""_);_(@_)", _
405                                          "_(* 0.0%_);_(* (0.0%);_(* ""- %""_);_(@_)", _
406                                          "_(* 0.00%_);_(* (0.00%);_(* ""- %""_);_(@_)"}
407
408     '''
409     ' Default constructor; initializes (opens) the output templates.
410     '
411     Public Sub New()
412         Me.OpenTemplates()
413     End Sub
414
415     '''
416     ' Opens the output template files.
417     '
418     Public Sub OpenTemplates()
419         With CAFE.TemplateSettings
420             If .UseTemplates Then
421                 If .DefaultTemplatePath <> "" Then
422                     Me.DefaultTemplate = New C_Cafe_InFile
423                     Me.DefaultTemplate.OpenFile(.DefaultTemplatePath)
424                 Else
425                     .UseTemplates = False
426                     Exit Sub
427                 End If
428
429                 'If .LandscapeTemplatePath <> "" Then
430                 '    Me.LandscapeTemplate = New C_Cafe_InFile
431                 '    Me.LandscapeTemplate.OpenFile(.LandscapeTemplatePath)
432                 'End If
433
434                 'If .PortraitTemplatePath <> "" Then
435                 '    Me.PortraitTemplate = New C_Cafe_InFile
436                 '    Me.PortraitTemplate.OpenFile(.PortraitTemplatePath)
437                 'End If
438             End If
439         End With
440     End Sub
441
442     '''
443     ' Closes the template files.
444     '
445     Public Sub CloseTemplates()
446         If CAFE.TemplateSettings.UseTemplates Then
447             If Not IsNothing(Me.DefaultTemplate) Then
448                 Me.DefaultTemplate.CloseFile()
449             End If
450
```

```
451             If Not IsNothing(Me.LandscapeTemplate) Then
452                 Me.LandscapeTemplate.CloseFile()
453             End If
454
455             If Not IsNothing(Me.PortraitTemplate) Then
456                 Me.PortraitTemplate.CloseFile()
457             End If
458         End If
459     End Sub
460
461
462  Industry report...
463
464
465  Manufacturers report...
466
467
468  Vehicles report...
469
470
471 #Region " Effects report... "
472
473     '''
474     ' This method generates the effects report...
475     '
476     Public Sub ReportEffects()
477         CAFE.Profiling.FunctionStart()
478
479         ' declare vars
480         Dim i, j, k As Integer      ' counters
481
482         Dim path As String
483
484         Dim nHeaderRows As Integer
485         Dim nHeaderCols As Integer
486         Dim nRows As Integer
487         Dim nCols As Integer
488
489         Dim nEffVals As New EffNValsStr
490         Dim EffSectsStart As New EffSectsStr
491
492         Dim nEffOnStack As Integer      ' stores the total number of stack entries for all MY's for effects
493         Dim nMfg As Integer
494
495         Dim EffStack As EffectStackClass
496         Dim MfgStack As ManufacturerStackClass
497         Dim IndStack As IndustryWideStackClass
498         Dim ColData() As Object
499
500         ' initialize vars
501         path = Me.path & "\Effects_Summary_Sn" & Me.Scenario & ".xls"
502         Me.file = New C_CAFE_OutFile(Nothing, path)
503
504         Me.nRowsToRemove = 0
505         ReDim Me.RowsToRemove(Me.nRowsToRemove - 1)
506
507         Me.nColsToRemove = 0
508         ReDim Me.ColsToRemove(Me.nColsToRemove - 1)
509
510         nEffVals.nMfgT = CAFE.Mfgs.Length + 1
511         nEffVals.nFuelT = 3       ' = 2 + 1, (1 for total)
512         nEffVals.nCrPl = 5
513         nEffVals.nM6T = 9         ' = 8 + 1, (1 for total)
514         nEffVals.nSafetyT = 18    ' = 17 + 1, (1 for total)
515
516         EffSectsStart.EmplStart = 3
517         EffSectsStart.ECStart = EffSectsStart.EmplStart + (2 + CInt(Not CAFE.NMNLIncluded)) * nEffVals.nMfgT + 2
```

```
518         EffSectsStart.EmnsStart = EffSectsStart.ECStart + 4 * nEffVals.nFuelT + 2
519         EffSectsStart.SftStart = EffSectsStart.EmnsStart + nEffVals.nM6T * (nEffVals.nCrPl + 4) + 2
520         If CAFE.SafetyIncluded Then
521             EffSectsStart.OaSCostStart = EffSectsStart.SftStart + 3 * nEffVals.nSafetyT + 2
522         Else
523             EffSectsStart.OaSCostStart = EffSectsStart.EmnsStart + nEffVals.nM6T * (nEffVals.nCrPl + 4) + 2
524         End If
525         EffSectsStart.DOaSCostStart = EffSectsStart.OaSCostStart + 17
526
527         nHeaderRows = 2
528         nHeaderCols = 2
529
530         ' all effects have the same amount of years on stack, so use the first stack entry
531         If IsNothing(CAFE.EffectsStack) Then
532             nEffOnStack = 0
533         Else
534             nEffOnStack = CAFE.EffectsStack(0).Size()
535         End If
536
537         nMfg = CAFE.Mfgs.Length
538
539         Me.ReportEffects_SetupRowsAndColsToRemove(nEffOnStack / 3)
540
541         nRows = 43 + (2 + CInt(Not CAFE.NMNLIncluded)) * nEffVals.nMfgT + 4 * nEffVals.nFuelT + (nEffVals.nCrPl + 4) * nEffVals.nM6T + 3 * nEffVals.nSafetyT
542         If Not CAFE.SafetyIncluded Then
543             nRows -= (3 * nEffVals.nSafetyT + 2)
544         End If
545         nCols = nHeaderCols + nEffOnStack
546
547         ' write output
548         Me.AddSummaryPage()
549
550         ' reset the buffer for the new run
551         Me.ResetBuffer(nRows, nCols)
552
553         ' write headers
554         Me.ReportEffects_SetupHeaders()
555
556         ' reset column counter to the first data column
557         Me.col = 2
558
559         ' loop through the stack to output each MY for this scenario
560         For i = 0 To nEffOnStack - 1
561             ' reset row counter to the first data row
562             Me.row = 0
563
564             ' write data
565             '
566             ' get the stacks
567             EffStack = CAFE.EffectsStack(Scenario).Read(i)
568
569             ' get the col data
570             ColData = EffStack.ToArray()
571
572             ' get the model year for this stack's entry
573             Me.MY = EffStack.MY
574
575             ' write model years headers
576             Me.WriteMYHeader(i Mod 3 = 0, Me.row, Me.col)
577
578             ' first, write the "employment" section
579             If CAFE.NMNLIncluded Then
580                 For j = 0 To nMfg - 1
581                     MfgStack = CAFE.MfgsStack(Scenario)(j).Read(i)
582                     Me.WriteCellV(MfgStack.TotalUSEmploymentJobs_Overall)
583                 Next j
```

```
584                 IndStack = CAFE.IndustriesStack(Scenario).Read(i)
585                 Me.WriteCellV(IndStack.TotalUSEmploymentJobs_Overall)
586             End If
587             '
588             For j = 0 To nMfg - 1
589                 MfgStack = CAFE.MfgsStack(Scenario)(j).Read(i)
590                 Me.WriteCellV(MfgStack.TotalSales_Overall)
591             Next j
592             IndStack = CAFE.IndustriesStack(Scenario).Read(i)
593             Me.WriteCellV(IndStack.TotalSales_Overall)
594             '
595             Me.row += 2     ' move on to the next row
596
597             ' next, write the "energey consumption" section
598             For j = 0 To 3
599                 Me.WriteCellsV(ColData(j))
600             Next j
601             Me.row += 2     ' move on to the next row
602
603             ' next, write the "emissions" section
604             For j = 4 To 12
605                 Me.WriteCellsV(ColData(j))
606             Next j
607             Me.row += 2     ' move on to the next row
608
609             ' next, write the "safety" section (only if RunSafetyCalcs is true)
610             If CAFE.SafetyIncluded Then
611                 For j = 13 To 15
612                     Me.WriteCellsV(ColData(j))
613                 Next j
614                 Me.row += 2     ' move on to the next row
615             End If
616
617             ' finally, write the "owner and societal costs" sections
618             ' undiscounted
619             For j = 16 To 30
620                 Me.WriteCellV(ColData(j))
621             Next j
622             Me.row += 2
623             '
624             ' discounted
625             For j = 31 To ColData.Length - 1
626                 Me.WriteCellV(ColData(j))
627             Next j
628
629             ' increment col counter to move on to the next data column
630             Me.col += 1
631
632             '' Record key info for a summary chart
633             'If (i Mod 3 = 0) Then
634             '    RecordCells(CAFE.SummaryOutput.SummaryTypeEnum.LifeTimeFuelConsumption,
        ColData(0), 2, Me.Scenario, Me.MY)
635             '    RecordCells(CAFE.SummaryOutput.SummaryTypeEnum.TotalLifeTimeFatalities,
        ColData(13), 17, Me.Scenario, Me.MY)
636             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscTotalFuelExpend,
        ColData(16), Me.Scenario, Me.MY)
637             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscFuelTaxRevenues,
        ColData(17), Me.Scenario, Me.MY)
638             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscTravelValue, ColData
        (18), Me.Scenario, Me.MY)
639             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscRefuelingTimeValue,
        ColData(19), Me.Scenario, Me.MY)
640             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.
        UndiscPetroMarketExternalities, ColData(20), Me.Scenario, Me.MY)
641             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscCongestionCosts,
        ColData(21), Me.Scenario, Me.MY)
642             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscNoiseCosts, ColData(22)
        , Me.Scenario, Me.MY)
643             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscAccidentCosts, ColData
```

```
(23), Me.Scenario, Me.MY)
644             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscCO2, ColData(24), Me.
Scenario, Me.MY)
645             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscCO, ColData(25), Me.
Scenario, Me.MY)
646             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscVOC, ColData(26), Me.
Scenario, Me.MY)
647             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscNOX, ColData(27), Me.
Scenario, Me.MY)
648             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscPM, ColData(28), Me.
Scenario, Me.MY)
649             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.UndiscSOX, ColData(29), Me.
Scenario, Me.MY)
650             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscTotalFuelExpend, ColData
(31), Me.Scenario, Me.MY)
651             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscFuelTaxRevenues, ColData
(32), Me.Scenario, Me.MY)
652             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscTravelValue, ColData(33),
 Me.Scenario, Me.MY)
653             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscRefuelingTimeValue,
ColData(34), Me.Scenario, Me.MY)
654             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscPetroMarketExternalities,
 ColData(35), Me.Scenario, Me.MY)
655             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscCongestionCosts, ColData
(36), Me.Scenario, Me.MY)
656             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscNoiseCosts, ColData(37),
Me.Scenario, Me.MY)
657             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscAccidentCosts, ColData
(38), Me.Scenario, Me.MY)
658             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscCO2, ColData(39), Me.
Scenario, Me.MY)
659             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscCO, ColData(40), Me.
Scenario, Me.MY)
660             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscVOC, ColData(41), Me.
Scenario, Me.MY)
661             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscNOX, ColData(42), Me.
Scenario, Me.MY)
662             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscPM, ColData(43), Me.
Scenario, Me.MY)
663             '    RecordCell(CAFE.SummaryOutput.SummaryTypeEnum.DiscSOX, ColData(44), Me.
Scenario, Me.MY)
664             'End If
665         Next i
666
667         ' now save data to excel
668         '
669         ' add a new worksheet
670         Me.AddWorksheet("Sn" & Me.Scenario, 2, CAFE.TemplateSettings.UseTemplates,
"TEMPLATE_EFF")
671
672         ' and write the buffer
673         Me.WriteBuffer(2, nRows, nCols)
674
675         If Not CAFE.TemplateSettings.UseTemplates Then
676             ' merge the header columns that need to be merged
677             Me.ReportEffects_MergeCells(nHeaderRows, nRows, nCols, EffSectsStart, nEffVals)
678
679             ' bold and center the header cells
680             Me.ReportEffects_BoldAndCenter(nHeaderRows, nHeaderCols, nRows, nCols)
681
682             ' format cells
683             Me.ReportEffects_FormatCells(nHeaderRows, nHeaderCols, nRows, nCols,
EffSectsStart, nEffVals)
684         End If
685
686         ' apply final formatting
687         Me.ReportEffects_Finalize(nHeaderRows, nHeaderCols, nRows, nCols, EffSectsStart,
nEffVals)
688
```

```
689         ' finalize the report
690         Me.FinalizeReport()
691
692         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
693     End Sub
694
695     Private Sub ReportEffects_SetupRowsAndColsToRemove(ByVal nMY As Integer)
696         CAFE.Profiling.FunctionStart()
697
698         Dim i As Integer
699         Dim nMfg As Integer
700
701         nMfg = CAFE.Mfgs.Length
702
703         If CAFE.TemplateSettings.UseTemplates Then
704             ' get columns to remove from the template
705             If nMY < 19 Then
706                 Me.nColsToRemove += 1
707                 ReDim Preserve Me.ColsToRemove(Me.nColsToRemove - 1)
708                 Me.ColsToRemove(Me.nColsToRemove - 1) = "3:" + (2 + 3 * (19 - nMY)).ToString ↙
    ()
709             End If
710
711             ' get rows to remove from the template
712             If nMfg < 18 Then
713                 Me.nRowsToRemove += 2
714                 ReDim Preserve Me.RowsToRemove(Me.nRowsToRemove - 1)
715
716                 ' Total U.S. Employment - Jobs
717                 If CAFE.NMNLIncluded Then
718                     Me.RowsToRemove(Me.nRowsToRemove - 2) = (4 + nMfg).ToString() + ":21"
719                 Else
720                     ' nmnl worksheet was not present--remove US employment - jobs
721                     Me.RowsToRemove(Me.nRowsToRemove - 2) = "4:22"
722                 End If
723
724                 ' Total Sales
725                 Me.RowsToRemove(Me.nRowsToRemove - 1) = (23 + nMfg).ToString() + ":40"
726             End If
727
728             If Not CAFE.SafetyIncluded Then
729                 Me.nRowsToRemove += 1
730                 ReDim Preserve Me.RowsToRemove(Me.nRowsToRemove - 1)
731                 Me.RowsToRemove(Me.nRowsToRemove - 1) = (140).ToString() + ":" + (195). ↙
    ToString()
732             End If
733         End If
734
735         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
736     End Sub
737
738     Private Sub ReportEffects_SetupHeaders()
739         CAFE.Profiling.FunctionStart()
740
741         ' declare vars
742         Dim i As Integer
743         Dim tmpRow As Integer           ' stores temporary row and cols values when
744         Dim tmoCol As Integer           ' transitioning between multiple sections
745
746         Dim nMfg As Integer
747         Dim nFuel As Integer
748         Dim nM6 As Integer
749         Dim nSafety As Integer
750         Dim nCrPol As Integer
751
752         ' initialize vars
753         Dim MfgList() As Object = AddTotal(GetMfgList())
754         Dim FuelList As Object = AddTotal(GetFuelList())
755         Dim Mobile6List() As Object = AddTotal(GetMobile6List())
```

```
        Dim SafetyList() As Object = AddTotal(GetSafetyList())
        Dim CriteriaPollutants() As Object = GetCriteriaPollutants()

        nMfg = MfgList.Length
        nFuel = FuelList.Length
        nM6 = Mobile6List.Length
        nSafety = SafetyList.Length
        nCrPol = CriteriaPollutants.Length

        ' write headers
        '
        ' first, write "employment" section
        Me.row = 2
        Me.col = 0
        Me.WriteCellV("Employment")
        If CAFE.NMNLIncluded Then
            Me.WriteCellV("Total U.S. Employment - Jobs", nMfg)
        End If
        Me.WriteCell("Total Sales")
        Me.row = 3
        For i = 1 + CInt(CAFE.NMNLIncluded) To 1
            Me.WriteCellsV(MfgList)
        Next

        ' next, write "energy consumption" section
        Dim Headers1() As Object = {"Lifetime Fuel Consumption (k gal.)", "Sales", "Lifetime VMT (k mi.)", "Average Fuel Economy (mpg)"}
        Me.row += 1     ' increment row counter by 1 to leave a free space between all effects sections
        Me.col = 0
        Me.WriteCellV("Energy Consumption")
        For i = 0 To Headers1.Length - 1
            Me.WriteCell(Headers1(i))
            Me.WriteCellsV(FuelList)
            Me.col = 0      ' reset col to the first column
        Next i

        ' next, write "emissions" section
        Me.row += 1
        Me.WriteCellV("Emissions")
        Me.WriteCell("CO2 (k MTCE)")
        Me.WriteCellsV(Mobile6List)
        Me.col = 0
        For i = 0 To nCrPol - 1
            Me.WriteCell(CriteriaPollutants(i) & " (tons)")
            Me.WriteCellsV(Mobile6List)
            Me.col = 0
        Next i
        For i = 0 To 2
            Me.WriteCell(Headers1(i + 1))
            Me.WriteCellsV(Mobile6List)
            Me.col = 0
        Next i

        ' next, write "safety" section
        If CAFE.SafetyIncluded Then
            Dim Headers2() As Object = {"Total Lifetime Fatalities", "Sales", "Average Curb Weight (lb.)"}
            'Dim Headers2() As Object = {"Total Lifetime Fatalities", "Sales", "Lifetime VMT (k mi.)", "Average Curb Weight (lb.)"}
            Me.row += 1
            Me.WriteCellV("Safety")
            For i = 0 To Headers2.Length - 1
                Me.WriteCell(Headers2(i))
                Me.WriteCellsV(SafetyList)
                Me.col = 0
            Next i
        End If

```

```
821         ' finally, write "owner and societal costs" sections
822         Me.ReportEffects_SetupHeaders_OwnerSocCosts("Undiscounted")
823         Me.ReportEffects_SetupHeaders_OwnerSocCosts("Discounted")
824
825         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
826     End Sub
827
828     Private Sub ReportEffects_SetupHeaders_OwnerSocCosts(ByVal DiscStr As String)
829         CAFE.Profiling.FunctionStart()
830
831         Dim i As Integer
832         Dim nCrPol As Integer
833
834         ' initialize vars
835         Dim CriteriaPollutants() As Object = GetCriteriaPollutants()
836         nCrPol = CriteriaPollutants.Length
837
838         Dim Headers3() As Object = {"Total Lifetime Pretax Fuel Expenditures", "Fuel Tax
    Revenues", _
839                                     "Travel Value", "Refueling Time Value", "Petroleum
    Market Externalities", _
840                                     "Congestion Costs", "Noise Costs", "Accident Costs",
    "CO2"}
841         Me.row += 1
842         Me.WriteCellV(DiscStr + " Owner and Societal Costs (k $)")
843         For i = 0 To Headers3.Length - 1
844             Me.WriteCellV(Headers3(i))
845         Next i
846         For i = 0 To nCrPol - 1
847             Me.WriteCellV(CriteriaPollutants(i))
848         Next i
849         Me.WriteCellV("[for future use]")       ' KG (11/23/04): was "Monetized Fatalities"
850         'Me.WriteCellV("Monetized Fatalities")
851
852         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
853     End Sub
854
855     Private Sub ReportEffects_MergeCells(ByVal nHeaderRows As Integer, _
856                                          ByVal nRows As Integer, _
857                                          ByVal nCols As Integer, _
858                                          ByVal EffSectsStart As EffSectsStr, _
859                                          ByVal nEffVals As EffNValsStr)
860         '
861         CAFE.Profiling.FunctionStart()
862
863         '
864         Dim i As Integer
865
866         Dim cell1 As Object
867         Dim cell2 As Object
868
869         ' merge data...
870         '
871         ' merge the header rows
872         cell1 = Me.range.Cells(1, 1)
873         cell2 = Me.range.Cells(2, 2)
874         Merge(Me.range.Range(cell1, cell2), , True)
875
876         For i = 3 To nCols - 1 Step 3
877             cell1 = Me.range(1, i)
878             cell2 = Me.range(1, i + 2)
879             Merge(Me.range.Range(cell1, cell2), , True)
880         Next i
881
882         ' first, merge "employment" section
883         cell1 = Me.range.Cells(EffSectsStart.EmplStart, 1)
884         cell2 = Me.range.Cells(EffSectsStart.EmplStart, 2)
885         Merge(Me.range.Range(cell1, cell2), , True)
886
```

```
887         For i = 0 To (2 + CInt(Not CAFE.NMNLIncluded)) * nEffVals.nMfgT - 1 Step nEffVals.
    nMfgT
888             cell1 = Me.range.Cells(EffSectsStart.EmplStart + 1 + i, 1)
889             cell2 = Me.range.Cells(EffSectsStart.EmplStart + 1 + i + nEffVals.nMfgT - 1, 1)
890             Merge(Me.range.Range(cell1, cell2), , True)
891         Next i
892
893         ' next, merge "energy consumption" section
894         cell1 = Me.range.Cells(EffSectsStart.ECStart, 1)
895         cell2 = Me.range.Cells(EffSectsStart.ECStart, 2)
896         Merge(Me.range.Range(cell1, cell2), , True)
897
898         For i = 0 To 4 * nEffVals.nFuelT - 1 Step nEffVals.nFuelT
899             cell1 = Me.range.Cells(EffSectsStart.ECStart + 1 + i, 1)
900             cell2 = Me.range.Cells(EffSectsStart.ECStart + 1 + i + nEffVals.nFuelT - 1, 1)
901             Merge(Me.range.Range(cell1, cell2), , True)
902         Next i
903
904         ' next, merge "emissions" section
905         cell1 = Me.range.Cells(EffSectsStart.EmnsStart, 1)
906         cell2 = Me.range.Cells(EffSectsStart.EmnsStart, 2)
907         Merge(Me.range.Range(cell1, cell2), , True)
908
909         For i = 0 To (4 + nEffVals.nCrPl) * nEffVals.nM6T - 1 Step nEffVals.nM6T
910             cell1 = Me.range.Cells(EffSectsStart.EmnsStart + 1 + i, 1)
911             cell2 = Me.range.Cells(EffSectsStart.EmnsStart + 1 + i + nEffVals.nM6T - 1, 1)
912             Merge(Me.range.Range(cell1, cell2), , True)
913         Next i
914
915         ' next, merge "safety" section
916         If CAFE.SafetyIncluded Then
917             cell1 = Me.range.Cells(EffSectsStart.SftStart, 1)
918             cell2 = Me.range.Cells(EffSectsStart.SftStart, 2)
919             Merge(Me.range.Range(cell1, cell2), , True)
920
921             For i = 0 To 3 * nEffVals.nSafetyT - 1 Step nEffVals.nSafetyT
922                 cell1 = Me.range.Cells(EffSectsStart.SftStart + 1 + i, 1)
923                 cell2 = Me.range.Cells(EffSectsStart.SftStart + 1 + i + nEffVals.nSafetyT -
    1, 1)
924                 Merge(Me.range.Range(cell1, cell2), , True)
925             Next i
926         End If
927
928         ' next, merge "undiscounted owner and societal costs" section
929         cell1 = Me.range.Cells(EffSectsStart.OaSCostStart, 1)
930         cell2 = Me.range.Cells(EffSectsStart.OaSCostStart, 2)
931         Merge(Me.range.Range(cell1, cell2), , True)
932
933         For i = 0 To 14
934             cell1 = Me.range.Cells(EffSectsStart.OaSCostStart + 1 + i, 1)
935             cell2 = Me.range.Cells(EffSectsStart.OaSCostStart + 1 + i, 2)
936             Merge(Me.range.Range(cell1, cell2), , True)
937         Next i
938
939         ' next, merge "discounted owner and societal costs" section
940         cell1 = Me.range.Cells(EffSectsStart.DOaSCostStart, 1)
941         cell2 = Me.range.Cells(EffSectsStart.DOaSCostStart, 2)
942         Merge(Me.range.Range(cell1, cell2), , True)
943
944         For i = 0 To 14
945             cell1 = Me.range.Cells(EffSectsStart.DOaSCostStart + 1 + i, 1)
946             cell2 = Me.range.Cells(EffSectsStart.DOaSCostStart + 1 + i, 2)
947             Merge(Me.range.Range(cell1, cell2), , True)
948         Next i
949
950         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
951     End Sub
952
953     Private Sub ReportEffects_BoldAndCenter(ByVal nHeaderRows As Integer, _
```

```
954                                             ByVal nHeaderCols As Integer, _
955                                             ByVal nRows As Integer, _
956                                             ByVal nCols As Integer)
957         '
958         CAFE.Profiling.FunctionStart()
959
960         '
961         ' since mfg reports are bolded and centered in the same way, use mfg's method
962         Me.ReportManufacturers_BoldAndCenter(nHeaderRows, nHeaderCols, nRows, nCols)
963
964         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
965     End Sub
966
967     Private Sub ReportEffects_FormatCells(ByVal nHeaderRows As Integer, _
968                                           ByVal nHeaderCols As Integer, _
969                                           ByVal nRows As Integer, _
970                                           ByVal nCols As Integer, _
971                                           ByVal EffSectsStart As EffSectsStr, _
972                                           ByVal nEffVals As EffNValsStr)
973         '
974         CAFE.Profiling.FunctionStart()
975
976         '
977         Dim i As Integer
978
979         Dim cell1 As Object
980         Dim cell2 As Object
981
982         ' apply cell formatting
983         '
984         ' first, format everything as text
985         cell1 = Me.range.Cells(1, 1)
986         cell2 = Me.range.Cells(nRows, nCols)
987         SetNumberFormat(Me.range.Range(cell1, cell2), "@")
988
989         ' next, format all data as comma-separated integer
990         cell1 = Me.range.Cells(nHeaderRows + 1, nHeaderCols + 1)
991         SetNumberFormat(Me.range.Range(cell1, cell2), Me.NumberFormats(0))
992
993         ' now we want to apply individual formatting for different sections
994         '
995         ' format fuel economy values as an integer with 2 decimal places
996         ' > energy consumption
997         i = 1 + EffSectsStart.ECStart + nEffVals.nFuelT * 3
998         cell1 = Me.range.Cells(i, nHeaderCols + 1)
999         cell2 = Me.range.Cells(i + 2, nCols)
1000        SetNumberFormat(Me.range.Range(cell1, cell2), Me.NumberFormats(2))
1001        ' > emissions
1002        i = 1 + EffSectsStart.EmnsStart + (nEffVals.nCrPl + 3) * nEffVals.nM6T
1003        cell1 = Me.range.Cells(i, nHeaderCols + 1)
1004        cell2 = Me.range.Cells(i + nEffVals.nM6T, nCols)
1005        SetNumberFormat(Me.range.Range(cell1, cell2), Me.NumberFormats(2))
1006
1007        ' format every 3rd column as %, since it is the "Delta (%)" column
1008        For i = nHeaderCols + 3 To nCols Step 3
1009            cell1 = Me.range.Cells(nHeaderRows + 1, i)
1010            cell2 = Me.range.Cells(nRows, i)
1011            SetNumberFormat(Me.range.Range(cell1, cell2), Me.NumberFormats(5))
1012        Next i
1013
1014        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1015    End Sub
1016
1017    Private Sub ReportEffects_Finalize(ByVal nHeaderRows As Integer, _
1018                                       ByVal nHeaderCols As Integer, _
1019                                       ByVal nRows As Integer, _
1020                                       ByVal nCols As Integer, _
1021                                       ByVal EffSectsStart As EffSectsStr, _
1022                                       ByVal nEffVals As EffNValsStr)
```

```
1023        '
1024        CAFE.Profiling.FunctionStart()
1025
1026        '
1027        Dim cell1 As Object
1028        Dim cell2 As Object
1029
1030        Dim rIdx As Integer
1031
1032        Me.FinalizeWorksheet(nHeaderRows, nHeaderCols, EffSectsStart.ECStart - 2, nCols, "C3 ↙
    ", "Effects Summary.", Color.Blue)
1033
1034        If Not CAFE.TemplateSettings.UseTemplates Then
1035            cell1 = Me.range.Cells(3, 1)
1036            cell2 = Me.range.Cells(3, nCols)
1037            HighlightCells(Me.range.Range(cell1, cell2), 36)
1038
1039            ' energy consumption
1040            cell1 = Me.range.Cells(EffSectsStart.ECStart, 1)
1041            cell2 = Me.range.Cells(EffSectsStart.ECStart, nCols)
1042            InsertPageBreak(cell1)
1043            HighlightCells(Me.range.Range(cell1, cell2), 36)
1044            Me.SetTableBorders(EffSectsStart.ECStart, 1, EffSectsStart.EmnsStart - 2, nCols, ↙
     Color.Blue)
1045            Me.SetTableBorders(EffSectsStart.ECStart, 1, EffSectsStart.EmnsStart - 2,       ↙
    nHeaderCols, Color.Blue, True)
1046
1047            ' emissions
1048            cell1 = Me.range.Cells(EffSectsStart.EmnsStart, 1)
1049            cell2 = Me.range.Cells(EffSectsStart.EmnsStart, nCols)
1050            InsertPageBreak(cell1)
1051            HighlightCells(Me.range.Range(cell1, cell2), 36)
1052            Me.SetTableBorders(EffSectsStart.EmnsStart, 1, EffSectsStart.SftStart - 2, nCols ↙
    , Color.Blue)
1053            Me.SetTableBorders(EffSectsStart.EmnsStart, 1, EffSectsStart.SftStart - 2,      ↙
    nHeaderCols, Color.Blue, True)
1054
1055            ' safety
1056            If CAFE.SafetyIncluded Then
1057                cell1 = Me.range.Cells(EffSectsStart.SftStart, 1)
1058                cell2 = Me.range.Cells(EffSectsStart.SftStart, nCols)
1059                InsertPageBreak(cell1)
1060                HighlightCells(Me.range.Range(cell1, cell2), 36)
1061                Me.SetTableBorders(EffSectsStart.SftStart, 1, EffSectsStart.OaSCostStart - 2 ↙
    , nCols, Color.Blue)
1062                Me.SetTableBorders(EffSectsStart.SftStart, 1, EffSectsStart.OaSCostStart - 2 ↙
    , nHeaderCols, Color.Blue, True)
1063            End If
1064
1065            ' undiscounted owner and societal costs
1066            cell1 = Me.range.Cells(EffSectsStart.OaSCostStart, 1)
1067            cell2 = Me.range.Cells(EffSectsStart.OaSCostStart, nCols)
1068            InsertPageBreak(cell1)
1069            HighlightCells(Me.range.Range(cell1, cell2), 36)
1070            Me.SetTableBorders(EffSectsStart.OaSCostStart, 1, EffSectsStart.DOaSCostStart -  ↙
    2, nCols, Color.Blue)
1071            Me.SetTableBorders(EffSectsStart.OaSCostStart, 1, EffSectsStart.DOaSCostStart -  ↙
    2, nHeaderCols, Color.Blue, True)
1072
1073            ' discounted owner and societal costs
1074            cell1 = Me.range.Cells(EffSectsStart.DOaSCostStart, 1)
1075            cell2 = Me.range.Cells(EffSectsStart.DOaSCostStart, nCols)
1076            InsertPageBreak(cell1)
1077            HighlightCells(Me.range.Range(cell1, cell2), 36)
1078            Me.SetTableBorders(EffSectsStart.DOaSCostStart, 1, nRows, nCols, Color.Blue)
1079            Me.SetTableBorders(EffSectsStart.DOaSCostStart, 1, nRows, nHeaderCols, Color.    ↙
    Blue, True)
1080
1081            ' separate sections with a solid horizontal line
```

```
1082            ' > employment
1083            rIdx = EffSectsStart.EmplStart + nEffVals.nMfgT
1084            ReportEffects_Finalize_SeparateSections(rIdx, rIdx + 1 * nEffVals.nMfgT,
    nEffVals.nMfgT, nCols, Color.Blue, 35)
1085            ' > energy consumption
1086            rIdx = EffSectsStart.ECStart + nEffVals.nFuelT
1087            ReportEffects_Finalize_SeparateSections(rIdx, rIdx + 3 * nEffVals.nFuelT,
    nEffVals.nFuelT, nCols, Color.Blue, 35)
1088            ' > emissions
1089            rIdx = EffSectsStart.EmnsStart + nEffVals.nM6T
1090            ReportEffects_Finalize_SeparateSections(rIdx, rIdx + 8 * nEffVals.nM6T, nEffVals
    .nM6T, nCols, Color.Blue, 35)
1091            ' > safety
1092            If CAFE.SafetyIncluded Then
1093                rIdx = EffSectsStart.SftStart + nEffVals.nSafetyT
1094                ReportEffects_Finalize_SeparateSections(rIdx, rIdx + 2 * nEffVals.nSafetyT,
    nEffVals.nSafetyT, nCols, Color.Blue, 35)
1095            End If
1096        End If
1097
1098        ' auto-fit all data to the worksheet
1099        cell1 = Me.range.Cells(1, 1)
1100        cell2 = Me.range.Cells(nRows, nCols)
1101        AutoFit(Me.range.Range(cell1, cell2))
1102
1103        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1104    End Sub
1105
1106    Private Sub ReportEffects_Finalize_SeparateSections(ByVal bIdx As Integer, _
1107                                                        ByVal eIdx As Integer, _
1108                                                        ByVal iStep As Integer, _
1109                                                        ByVal nCols As Integer, _
1110                                                        ByVal BorderColor As Object, _
1111                                                        ByVal HiglightColorIdx As Integer)
1112        '
1113        CAFE.Profiling.FunctionStart()
1114
1115        '
1116        Dim i As Integer
1117
1118        Dim cell1 As Object
1119        Dim cell2 As Object
1120
1121        For i = bIdx To eIdx Step iStep
1122            cell1 = Me.range.Cells(i, 2)
1123            cell2 = Me.range.Cells(i, nCols)
1124            SetFontBold(Me.range.Range(cell1, cell2))
1125            If i < eIdx Then
1126                SetBorder(Me.range.Range(cell1, cell2), Excel.XlBordersIndex.xlEdgeBottom, _
1127                          Excel.XlLineStyle.xlContinuous, , BorderColor)
1128            End If
1129            HighlightCells(Me.range.Range(cell1, cell2), HiglightColorIdx)
1130        Next i
1131
1132        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1133    End Sub
1134
1135    '''
1136    ' This function adds "total" to the array.
1137    '
1138    Private Function AddTotal(ByRef arr() As Object) As Object()
1139        Dim len As Integer = arr.Length
1140        ReDim Preserve arr(len)
1141        arr(len) = "Total"
1142        Return arr
1143    End Function
1144
1145    '''
1146    ' This function writes the model years and scenario headers--the first two rows of each
```

```
    effects
    ' section.
    '
    Private Function WriteMYHeader(ByVal Condition As Boolean, _
                                   ByVal StartingRow As Integer, _
                                   Optional ByVal StartingCol As Integer = 3) As Object
        '
        If Condition Then
            Me.row = StartingRow
            Me.col = StartingCol

            WriteCellV("MY " & Me.MY)
            WriteCell("Current" & Chr(10) & "Scenario")
            WriteCell("Delta" & Chr(10) & "(abs.)")
            WriteCell("Delta" & Chr(10) & "(%)")
        End If

        Me.row = StartingRow + 3
        Me.col = StartingCol
    End Function

#End Region


 Environmental Assessment report...


 Summary Page


    '''
    ' This function computes the technology utilization/applicability for a vehicle on the given
    ' vehicle stack.
    '
    Private Function GetTechUtilizationApplicability(ByRef VehStack As VehicleStackClass) As Object()
        Dim i, j As Integer

        Dim TechsStack As TechnologiesStackClass
        Dim CurrentTechStack As TechnologyStackClass
        Dim PreviousTechStack As TechnologyStackClass
        Dim UtilizationApplicability() As Object

        ' setup vehicle's tech info for output
        TechsStack = VehStack.Utilization
        ReDim UtilizationApplicability(VehStack.Applicability.Length - 1)

        For i = 0 To VehStack.Applicability.Length - 1
            ' if tech is not available, mark it with an x
            If Not VehStack.Applicability(i) Then
                UtilizationApplicability(i) = "x"
            End If

            If TechsStack.Size() > 0 Then
                PreviousTechStack = TechsStack.Read(0)
            Else
                PreviousTechStack = Nothing
            End If

            ' check to see if this technology is applied to the given vehicle initially
            If Not IsNothing(PreviousTechStack) AndAlso PreviousTechStack.TechUsed(i) Then
                UtilizationApplicability(i) = "-"
            End If

            For j = 1 To TechsStack.Size() - 1
                CurrentTechStack = TechsStack.Read(j)

                If PreviousTechStack.TechUsed(i) And Not CurrentTechStack.TechUsed(i) Then
```

```
1213                    UtilizationApplicability(i) += "+"
1214                ElseIf Not PreviousTechStack.TechUsed(i) And CurrentTechStack.TechUsed(i)
    Then
1215                    UtilizationApplicability(i) = "+"
1216                End If
1217
1218                PreviousTechStack = CurrentTechStack
1219            Next j
1220        Next i
1221
1222        Return UtilizationApplicability
1223    End Function
1224
1225    '''
1226    ' This function returns an array of all technology header titles.
1227    '
1228    Private Function GetTechHeaders(Optional ByVal noCR As Boolean = False) As Object()
1229        Dim i As Integer
1230        Dim TH() As Object = GetTechAbbr()
1231
1232        For i = 0 To CAFE.Technologies.GetNumTechnologies - 1
1233            Dim sp As String = Chr(10).ToString
1234            If noCR Then
1235                sp = ". "
1236            End If
1237            TH(i) = CStr(i + 1) & sp & TH(i)
1238        Next i
1239
1240        Return TH
1241    End Function
1242
1243    '''
1244    ' This function returns an array of all technology abbreviations.
1245    '
1246    Private Function GetTechAbbr() As Object()
1247        Dim i As Integer
1248        Dim TH(CAFE.Technologies.GetNumTechnologies - 1) As Object
1249
1250        For i = 0 To CAFE.Technologies.GetNumTechnologies - 1
1251            TH(i) = CAFE.Technologies.VehicleClasses(0).Technology(i).Abbr
1252        Next i
1253
1254        Return TH
1255    End Function
1256
1257    '''
1258    ' This function returns an array of all technology names.
1259    '
1260    Public Function GetTechNames() As Object()
1261        Dim i As Integer
1262        Dim TH(CAFE.Technologies.GetNumTechnologies - 1) As Object
1263
1264        For i = 0 To CAFE.Technologies.GetNumTechnologies - 1
1265            TH(i) = CAFE.Technologies.VehicleClasses(0).Technology(i).Name
1266        Next i
1267
1268        Return TH
1269    End Function
1270
1271    '''
1272    ' This function returns an array of manufactuer names.
1273    '
1274    Private Function GetMfgList() As Object()
1275        Dim i As Integer
1276        Dim nMfg As Integer = CAFE.Mfgs.Length
1277        Dim MfgList(nMfg - 1) As Object
1278
1279        For i = 0 To nMfg - 1
1280            MfgList(i) = CAFE.Mfgs(i).MfgName
```

```
1281        Next i
1282
1283        Return MfgList
1284    End Function
1285
1286    '''
1287    ' This function returns an array for the gas-deisel fuel list.
1288    '
1289    Private Function GetFuelList() As Object()
1290        Dim FuelList() = {"Gas", "Diesel"}
1291        Return FuelList
1292    End Function
1293
1294    '''
1295    ' This function returns an array of Mobile6 Class names.
1296    '
1297    Private Function GetMobile6List() As Object()
1298        Dim Mobile6List() As Object = {"LDDV", "LDGV", "LDDT1", "LDGT1", "LDDT2", "LDGT2",
    "HDDV2b", "HDGV2b"}
1299        Return Mobile6List
1300    End Function
1301
1302    '''
1303    ' This function returns an array of Safety Class names.
1304    '
1305    Private Function GetSafetyList() As Object()
1306        Dim SafetyList() As Object = {"Very Small Cars", "Small Cars", "Midsize Cars",
    "Large Cars", _
1307                                      "Sporty 2Dr Cars", "Muscle 2Dr Cars", "Economy 2Dr
    Cars", _
1308                                      "Other 2Dr Cars", "Small SUVs", "Midsize SUVs", "Large
     SUVs", _
1309                                      "Other 2Dr SUVs", "Compact Pickup", "Large Pickup",
    "Heavy Pickup", _
1310                                      "Minivan", "Large Van"}
1311        Return SafetyList
1312    End Function
1313
1314    '''
1315    ' This function returns an array of Criteria Pollutants names.
1316    '
1317    Private Function GetCriteriaPollutants() As Object()
1318        Dim CriteriaPollutants() As Object = {"CO", "VOC", "NOX", "PM", "SOX"}
1319        Return CriteriaPollutants
1320    End Function
1321
1322    '''
1323    ' This method writes a given value to the buffer.
1324    '
1325    ' Params:
1326    '   val             - the value to write to the buffer
1327    '   colIncrement    - the amount by which to increment the column counter, so that the
    buffer
1328    '                     automatically moves on to the next cell; default is 1
1329    '
1330    Private Sub WriteCell(ByVal val As Object, Optional ByVal colIncrement As Integer = 1)
1331        Me.WriteCell(val, 0, colIncrement)
1332    End Sub
1333
1334    Private Sub WriteCellV(ByVal val As Object, Optional ByVal rowIncrement As Integer = 1)
1335        Me.WriteCell(val, rowIncrement, 0)
1336    End Sub
1337
1338    Private Sub WriteCell(ByVal val As Object, ByVal rowIncrement As Integer, ByVal
    colIncrement As Integer)
1339        ' ---+---+---+---+--- DEBUG STUFF ---+---+---+---+--- '
1340        'If IsNumeric(val) Then
1341        '    If Double.IsNaN(val) Then
1342        '        val = "NaN#"
```

```
1343        '    ElseIf Double.IsNegativeInfinity(val) Then
1344        '        val = "-Inf#"
1345        '    ElseIf Double.IsNegativeInfinity(val) Then
1346        '        val = "+Inf#"
1347        '    ElseIf val = Double.Epsilon Then
1348        '        val = ""     ' "Not Implemented"
1349        '    End If
1350        'End If
1351
1352        If IsNothing(val) Then
1353            val = ""
1354        ElseIf IsNumeric(val) AndAlso (Double.IsNaN(val) OrElse Double.IsInfinity(val)) Then
1355            val = ""
1356        End If
1357
1358        Me.buffer(Me.row, Me.col) = val
1359        Me.col += colIncrement
1360        Me.row += rowIncrement
1361    End Sub
1362
1363    '''
1364    ' This method writes the given array of values to the buffer.  This method invokes the
1365    ' WriteCell() method when writing individual values of the values() array.
1366    '
1367    Private Sub WriteCells(ByVal values As System.Array)
1368        Me.WriteCells(values, True)
1369    End Sub
1370
1371    Private Sub WriteCellsV(ByVal values As System.Array)
1372        Me.WriteCells(values, False)
1373    End Sub
1374
1375    Private Sub WriteCells(ByVal values As System.Array, ByVal horizontall As Boolean)
1376        Dim i, j As Integer
1377        Dim obj() As Object
1378
1379        If horizontall Then
1380            For i = 0 To values.Length - 1
1381                If IsArray(values(i)) Then
1382                    WriteCells(values(i))
1383                Else
1384                    WriteCell(values(i))
1385                End If
1386            Next i
1387        Else
1388            For i = 0 To values.Length - 1
1389                If IsArray(values(i)) Then
1390                    WriteCellsV(values(i))
1391                Else
1392                    WriteCellV(values(i))
1393                End If
1394            Next i
1395        End If
1396    End Sub
1397
1398    Private Sub RecordCells(ByVal Info As SummaryInfoClass.SummaryTypeEnum, ByVal values As ↙
    System.Array, ByVal Idx As Integer, _
1399                            ByVal iScenario As Integer, ByVal MYear As Integer, Optional ↙
    ByVal iManufacturer As Integer = -1)
1400        If Idx = Integer.MaxValue Then Idx = values.Length - 1
1401        RecordCell(Info, values(Idx), iScenario, MYear, iManufacturer)
1402    End Sub
1403
1404    Private Sub RecordCell(ByVal Info As SummaryInfoClass.SummaryTypeEnum, ByVal Value As ↙
    Object, _
1405                           ByVal iScenario As Integer, ByVal MYear As Integer, _
1406                           Optional ByVal iManufacturer As Integer = -1)
1407        Dim mfgname As String = String.Empty
1408        If iManufacturer >= 0 Then
```

```
1409            mfgname = CAFE.Mfgs(iManufacturer).MfgName
1410        End If
1411        CAFE.SummaryOutput.RecordSummary(Info, iScenario, MYear, iManufacturer, Value)
1412    End Sub
1413    '''
1414    ' This method resets the buffer and the row and col indexes so that the data can be
1415    ' written again.
1416    '
1417    Private Sub ResetBuffer(ByVal nRows As Integer, ByVal nCols As Integer)
1418        ReDim Me.buffer(nRows, nCols)
1419        Me.row = 0
1420        Me.col = 0
1421    End Sub
1422
1423    '''
1424    ' This method moves the buffer on to the next row by incrementing the row counter and resetting
1425    ' the column counter to 0 or to the given colIdx.
1426    '
1427    ' Params:
1428    '   colIdx          - the initial column index, from which the buffer should start writing on
1429    '                     the next row; default is 0; if colIdx is set to less than 0, then the
1430    '                     default value is used.
1431    '
1432    Private Sub NewRow(Optional ByVal colIdx As Integer = 0)
1433        row += 1
1434        col = colIdx
1435    End Sub
1436
1437    '''
1438    ' Adds a summary page to the report by copying it from the Output_Summary_SnXY.xls file.
1439    '
1440    Private Sub AddSummaryPage()
1441        Dim SumFile As New C_Cafe_InFile
1442        SumFile.OpenFile(Me.SummaryPath)
1443
1444        ' copy summary worksheet and remove not needed worksheets
1445        CopyWorksheet(SumFile.GetWorkbookObject(), Me.file.GetWorkbookObject(), "Summary Page")
1446        RemoveAllWorksheets()
1447        NameWorkSheet(1, "Summary Page")
1448
1449        SumFile.CloseFile()
1450
1451        ' process gui events
1452        Application.DoEvents()
1453    End Sub
1454
1455    Private Sub AddWorksheet(ByVal WorksheetName As String, _
1456                             ByVal ActSheetIdx As Integer, _
1457                             ByVal UseTemplatess As Boolean, _
1458                             Optional ByVal TemplateName As String = "CAFE_TEMPLATE")
1459        '
1460        If UseTemplatess Then
1461            Dim src As Excel.Workbook = Me.DefaultTemplate.GetWorkbookObject()
1462            Dim dst As Excel.Workbook = Me.file.GetWorkbookObject()
1463
1464            ' copy template worksheet
1465            CopyWorksheet(src, dst, ActSheetIdx, TemplateName)
1466            NameWorkSheet(ActSheetIdx, WorksheetName)
1467        Else
1468            xlUtilities.AddWorkSheet(WorksheetName)
1469        End If
1470
1471        ' process gui events
1472        Application.DoEvents()
1473    End Sub
```

```
1474
1475        Private Sub WriteBuffer(ByVal ActSheetIdx As Integer, _
1476                                ByVal nRows As Integer, _
1477                                ByVal nCols As Integer, _
1478                                Optional ByVal RowOffset As Integer = 0, _
1479                                Optional ByVal ColOffset As Integer = 0)
1480            '
1481            Dim i As Integer
1482            Dim ActSheet As Excel.Worksheet
1483            Dim rn0 As Excel.Range
1484            Dim rn1 As Excel.Range
1485            Dim cell1 As Object
1486            Dim cell2 As Object
1487
1488            ActSheet = file.SetSheet(ActSheetIdx)
1489            rn0 = ActSheet.Cells(1, 1)
1490            rn1 = ActSheet.Cells(nRows, nCols)
1491            Me.range = ActSheet.Range(rn0, rn1)
1492
1493            ' remove the unneeded rows from the template
1494            For i = Me.nRowsToRemove - 1 To 0 Step -1
1495                xlUtilities.RemoveRows(Me.range, Me.RowsToRemove(i))
1496            Next i
1497
1498            ' remove the unneeded cols from the template
1499            For i = Me.nColsToRemove - 1 To 0 Step -1
1500                xlUtilities.RemoveCols(Me.range, Me.ColsToRemove(i))
1501            Next i
1502
1503            cell1 = Me.range.Cells(1 + RowOffset, 1 + ColOffset)
1504            cell2 = Me.range.Cells(nRows, nCols)
1505            Me.range.Range(cell1, cell2).Value() = Me.buffer
1506
1507            ' process gui events
1508            Application.DoEvents()
1509        End Sub
1510
1511        '''
1512        ' This method finalizes the worksheet by setting the header and the footer for the active
1513        ' excel worksheet.  This method also sets the worksheet title, draws borders, sets up the
1514        ' panes, and set the title rows.
1515        '
1516        ' Params:
1517        '   nHeaderRows     - the number of header rows on this worksheet; this parameter determines
1518        '                     the way that the borders will be drawn and identifies the position of the
1519        '                     title rows
1520        '   nRows           - total number of rows on the active worksheet
1521        '   nCols           - total number of columns on the active worksheet
1522        '   FreezePanesCell - the cell around which to freeze the panes
1523        '   WorksheetHeader - specifies the title that will be displayed on the worksheet header
1524        '   Color           - the color to use for the borders
1525        '
1526        Private Sub FinalizeWorksheet(ByVal nHeaderRows As Integer, _
1527                                      ByVal nHeaderCols As Integer, _
1528                                      ByVal nRows As Integer, _
1529                                      ByVal nCols As Integer, _
1530                                      ByVal FreezePanesCell As Object, _
1531                                      ByVal WorksheetHeader As String, _
1532                                      ByVal Color As Object)
1533            '
1534            Dim cell1 As Object
1535            Dim cell2 As Object
1536            Dim sRow As Integer = 1
1537
1538            ' process gui events
```

```
1539        Application.DoEvents()
1540
1541        ' Page setup and setting the footer is done in the template.
1542        If Not CAFE.TemplateSettings.UseTemplates Then
1543            ' if we are not using templates, than do page setup
1544            '
1545            ' set up the page as landscape with .5 inch left and right margins,
1546            ' with .75 top and bottom margins, and fit it to 1 page wide
1547            PageSetup(Excel.XlPageOrientation.xlLandscape, _
1548                      Margin._0_50_INCH, Margin._0_50_INCH, _
1549                      Margin._0_75_INCH, Margin._0_75_INCH, 1)
1550            ' set up footer
1551            SetWorksheetFooter()
1552        ElseIf Me.ReportType = ReportTypeEnum.VEHICLES Then
1553            sRow = nHeaderRows + 1
1554        End If
1555
1556        ' process gui events
1557        Application.DoEvents()
1558
1559        ' set up header
1560        If CAFE.TemplateSettings.UseTemplates AndAlso (Me.ReportType = ReportTypeEnum.
    SUMMARY OrElse _
1561                                                       Me.ReportType = ReportTypeEnum.
    INDUSTRY OrElse _
1562                                                       Me.ReportType = ReportTypeEnum.
    EFFECTS) Then
1563            '
1564            SetWorksheetHeader("")
1565        Else
1566            SetWorksheetHeader(WorksheetHeader)
1567        End If
1568
1569        ' process gui events
1570        Application.DoEvents()
1571
1572        ' set table borders
1573        If Not CAFE.TemplateSettings.UseTemplates OrElse (Me.ReportType = ReportTypeEnum.
    VEHICLES OrElse _
1574                                                          Me.ReportType = ReportTypeEnum.
    EAS) Then
1575            '
1576            Me.SetTableBorders(sRow, 1, nRows, nCols, Color)
1577            Me.SetTableBorders(sRow, 1, nRows, nHeaderCols, Color, True)
1578        End If
1579
1580        ' process gui events
1581        Application.DoEvents()
1582
1583        If Not CAFE.TemplateSettings.UseTemplates Then
1584            ' set solid thin borders in blue for the header rows, if any
1585            If nHeaderRows <> 0 Then
1586                cell1 = Me.range.Cells(1, 1)
1587                cell2 = Me.range.Cells(nHeaderRows, nCols)
1588                SetBorders(Me.range.Range(cell1, cell2), , , Color)
1589            End If
1590            ' set solid medium outer border in specified color around the header row(s) (to
    make the header look framed)
1591            SetBorder(Me.range.Range(cell1, cell2), Excel.XlBordersIndex.xlEdgeLeft, , Excel
    .XlBorderWeight.xlMedium, Color)
1592            SetBorder(Me.range.Range(cell1, cell2), Excel.XlBordersIndex.xlEdgeRight, ,
    Excel.XlBorderWeight.xlMedium, Color)
1593            SetBorder(Me.range.Range(cell1, cell2), Excel.XlBordersIndex.xlEdgeTop, , Excel.
    XlBorderWeight.xlMedium, Color)
1594            SetBorder(Me.range.Range(cell1, cell2), Excel.XlBordersIndex.xlEdgeBottom, ,
    Excel.XlBorderWeight.xlMedium, Color)
1595        End If
1596
1597        ' process gui events
```

```
1598        Application.DoEvents()
1599
1600        ' auto-fit all data to the worksheet
1601        cell1 = Me.range.Cells(1, 1)
1602        cell2 = Me.range.Cells(nRows, nCols)
1603        AutoFit(Me.range.Range(cell1, cell2))
1604
1605        If Not CAFE.TemplateSettings.UseTemplates Then
1606            ' freeze panes
1607            FreezePanes(Me.range, FreezePanesCell)
1608            ' repeat the first two (or three) rows on each page
1609            PrintTitleRows(1, nHeaderRows)
1610        End If
1611
1612        ' process gui events
1613        Application.DoEvents()
1614    End Sub
1615
1616    '''
1617    ' This method sets up the border for a table specified by the given start and end rows and cols.
1618    '
1619    Private Sub SetTableBorders(ByVal sRow As Integer, _
1620                                ByVal sCol As Integer, _
1621                                ByVal eRow As Integer, _
1622                                ByVal eCol As Integer, _
1623                                ByVal Color As Object, _
1624                                Optional ByVal SolidBorder As Boolean = False)
1625        '
1626        Dim cell1 As Object
1627        Dim cell2 As Object
1628
1629        ' set dotted thin border in the specified color
1630        cell1 = Me.range.Cells(sRow, sCol)
1631        cell2 = Me.range.Cells(eRow, eCol)
1632        If SolidBorder Then
1633            SetBorders(Me.range.Range(cell1, cell2), Excel.XlLineStyle.xlContinuous, , Color)
1634        Else
1635            SetBorders(Me.range.Range(cell1, cell2), Excel.XlLineStyle.xlDash, , Color)
1636        End If
1637        ' set solid medium outer border in specified color for the whole table
1638        SetBorder(Me.range.Range(cell1, cell2), Excel.XlBordersIndex.xlEdgeLeft, , Excel.XlBorderWeight.xlMedium, Color)
1639        SetBorder(Me.range.Range(cell1, cell2), Excel.XlBordersIndex.xlEdgeRight, , Excel.XlBorderWeight.xlMedium, Color)
1640        SetBorder(Me.range.Range(cell1, cell2), Excel.XlBordersIndex.xlEdgeTop, , Excel.XlBorderWeight.xlMedium, Color)
1641        SetBorder(Me.range.Range(cell1, cell2), Excel.XlBordersIndex.xlEdgeBottom, , Excel.XlBorderWeight.xlMedium, Color)
1642    End Sub
1643
1644    '''
1645    ' This method is common for vehicle, manufacturer, and industry-wide reports.
1646    '
1647    ' ReportFinalize() must be called at the end of each report-generating method,
1648    '                  or when an exception is thrown before a report is completly
1649    '                  generated, to ensure that the necessary clean-up steps take
1650    '                  place
1651    '
1652    Public Sub FinalizeReport(Optional ByVal CloseFile As Boolean = True)
1653        Me.range = Nothing
1654        If CloseFile Then
1655            If Not IsNothing(Me.file) Then
1656                Me.file.CloseFile()
1657            End If
1658            Me.file = Nothing
1659        End If
1660        Me.ResetBuffer(0, 0)
```

```
1661            Me.buffer = Nothing
1662            Me.MY = 0
1663        End Sub
1664
1665        '''
1666        ' Returns the reports buffer.
1667        '
1668        Public Function GetBuffer() As Object(,)
1669            Return buffer
1670        End Function
1671
1672        '''
1673        ' Prints out the current contents of the buffer.
1674        '
1675        Public Sub PrintBuffer(Optional ByVal title As String = "")
1676            Dim i, j
1677
1678            Dim OutputFile As StreamWriter
1679            Dim fname As String = CAFE.OutputSettings.DefaultOutputPath & "\Scenario_" & Me.
    Scenario & _
1680                                  "\" & title & "__buffer__Sn" & Me.Scenario & ".csv"
1681
1682            OutputFile = New StreamWriter(fname)
1683
1684            For i = 0 To buffer.GetUpperBound(0)
1685                For j = 0 To buffer.GetUpperBound(1)
1686                    OutputFile.Write(buffer(i, j) & ",")
1687                Next j
1688                OutputFile.WriteLine()
1689            Next i
1690
1691            OutputFile.Close()
1692        End Sub
1693
1694 End Class
1695
```

```
Public Class ParametersClass


 Compliance Model Data


#Region " Effects Model Data "

    'set up vehicle age data (26 rows and 1 column) vars ... into arrays
    'average annual miles driven by surviving vehicles (0 to 25 years)
    Public milesAuto(25, 0) As Double
    Public milesPickup(25, 0) As Double
    Public milesVan(25, 0) As Double
    Public milesSuv(25, 0) As Double
    Public milesLTDefault(25, 0) As Double

    ' miles driven by Mobile6 clases (not used?)
    '''Public milesLDGV(25, 0) As Double
    '''Public milesLDDV(25, 0) As Double
    '''Public milesLDGT1(25, 0) As Double
    '''Public milesLDGT2(25, 0) As Double
    '''Public milesLDDV1(25, 0) As Double
    '''Public milesLDDV2(25, 0) As Double
    '''Public milesHDGV2b(25, 0) As Double
    '''Public milesHDDV2b(25, 0) As Double

    'proportion of original sales surviving (0 to 25 years).
    Public survivalLDGV(25, 0) As Double
    Public survivalLDDV(25, 0) As Double
    Public survivalLDGT1(25, 0) As Double
    Public survivalLDGT2(25, 0) As Double
    Public survivalLDDV1(25, 0) As Double
    Public survivalLDDV2(25, 0) As Double
    Public survivalHDGV2b(25, 0) As Double
    Public survivalHDDV2b(25, 0) As Double

    'set up fuel properties var
    'energy density (BTU/gal)
    Public convGasEngDens As Double
    Public fedGasEngDens As Double
    Public caliGasEngDens As Double
    Public convDiesEngDens As Double
    Public lowDiesEngDens As Double
    'mass density (gram/gal)
    Public convGasMassDens As Double
    Public fedGasMassDens As Double
    Public caliGasMassDens As Double
    Public convDiesMassDens As Double
    Public lowDiesMassDens As Double
    'carbon content (%)
    Public convGasCarb As Double
    Public fedGasCarb As Double
    Public caliGasCarb As Double
    Public convDiesCarb As Double
    Public lowDiesCarb As Double
    'sulfur content (ppm by weight)
    Public convGasSulf As Double
    Public fedGasSulf As Double
    Public caliGasSulf As Double
    Public convDiesSulf As Double
    Public lowDiesSulf As Double
    Public gasShareImp As Double                    'Share of Base-Case Fuel Use Imported as
    Refined Fuel
    Public gasShareDomRef As Double                 'Share of Base-Case Fuel Use Refined
    within U.S.
    Public gasShareDomRefDomCrude As Double         'Share Refined from Domestic Crude
    Public gasShareDomRefImpCrude As Double         'Share Refined from Imported Crude
    Public gasLowImp As Double                      'Share of Fuel Savings Leading to Lower
    Fuel Imports
```

```
67      Public gasLowDomRef As Double                   'Share of Fuel Savings Leading to Reduced
         Domestic Fuel Refining
68      Public gasLowDomRefDomCrude As Double           'Share of Reduced Domestic Refining from
        Domestic Crude
69      Public gasLowDomRefImpCrude As Double           'Share of Reduced Domestic Refining from
        Imported Crude
70      Public diesShareImp As Double                   'Share of Base-Case Fuel Use Imported as
        Refined Fuel
71      Public diesShareDomRef As Double                'Share of Base-Case Fuel Use Refined
        within U.S.
72      Public diesShareDomRefDomCrude As Double        'Share Refined from Domestic Crude
73      Public diesShareDomRefImpCrude As Double        'Share Refined from Imported Crude
74      Public diesLowImp As Double                     'Share of Fuel Savings Leading to Lower
        Fuel Imports
75      Public diesLowDomRef As Double                  'Share of Fuel Savings Leading to Reduced
         Domestic Fuel Refining
76      Public diesLowDomRefDomCrude As Double          'Share of Reduced Domestic Refining from
        Domestic Crude
77      Public diesLowDomRefImpCrude As Double          'Share of Reduced Domestic Refining from
        Imported Crude
78      'assumed duel mix (%)
79      Public convGasMix As Double                     'conventional gas
80      Public fedGasMix As Double                      'federal reformulated gas
81      Public fedNorthGasMix As Double                 'northern federal reformulated gas
82      Public fedSouthGasMix As Double                 'southern federal reformulated gas
83      Public caliGasMix As Double                     'california gas
84      Public convDiesMix As Double                    'conventional diesel
85      Public lowDiesMix As Double                     'low sulfur diesel
86
87      ' set up upstream emission vars
88      ' conv = conventional
89      ' fed = federal reformulated
90      ' cali = california
91      ' Total Emissions by Stage of Fuel Production and Distribution (grams/million Btu):
        petroleum extracted
92      Public totExtrConvGasCO As Double
93      Public totExtrFedGasCO As Double
94      Public totExtrCaliGasCO As Double
95      Public totExtrConvDiesCO As Double
96      Public totExtrLowDiesCO As Double
97      Public totExtrConvGasVOC As Double
98      Public totExtrFedGasVOC As Double
99      Public totExtrCaliGasVOC As Double
100     Public totExtrConvDiesVOC As Double
101     Public totExtrLowDiesVOC As Double
102     Public totExtrConvGasNOX As Double
103     Public totExtrFedGasNOX As Double
104     Public totExtrCaliGasNOX As Double
105     Public totExtrConvDiesNOX As Double
106     Public totExtrLowDiesNOX As Double
107     Public totExtrConvGasSOX As Double
108     Public totExtrFedGasSOX As Double
109     Public totExtrCaliGasSOX As Double
110     Public totExtrConvDiesSOX As Double
111     Public totExtrLowDiesSOX As Double
112     Public totExtrConvGasPM As Double
113     Public totExtrFedGasPM As Double
114     Public totExtrCaliGasPM As Double
115     Public totExtrConvDiesPM As Double
116     Public totExtrLowDiesPM As Double
117     Public totExtrConvGasCO2 As Double
118     Public totExtrFedGasCO2 As Double
119     Public totExtrCaliGasCO2 As Double
120     Public totExtrConvDiesCO2 As Double
121     Public totExtrLowDiesCO2 As Double
122     Public totExtrConvGasGHG As Double
123     Public totExtrFedGasGHG As Double
124     Public totExtrCaliGasGHG As Double
125     Public totExtrConvDiesGHG As Double
```

```
126     Public totExtrLowDiesGHG As Double
127     Public totTransConvGasCO As Double
128     ' Total Emissions by Stage of Fuel Production and Distribution (grams/million Btu):
        petroleum transportation
129     Public totTransFedGasCO As Double
130     Public totTransCaliGasCO As Double
131     Public totTransConvDiesCO As Double
132     Public totTransLowDiesCO As Double
133     Public totTransConvGasVOC As Double
134     Public totTransFedGasVOC As Double
135     Public totTransCaliGasVOC As Double
136     Public totTransConvDiesVOC As Double
137     Public totTransLowDiesVOC As Double
138     Public totTransConvGasNOX As Double
139     Public totTransFedGasNOX As Double
140     Public totTransCaliGasNOX As Double
141     Public totTransConvDiesNOX As Double
142     Public totTransLowDiesNOX As Double
143     Public totTransConvGasSOX As Double
144     Public totTransFedGasSOX As Double
145     Public totTransCaliGasSOX As Double
146     Public totTransConvDiesSOX As Double
147     Public totTransLowDiesSOX As Double
148     Public totTransConvGasPM As Double
149     Public totTransFedGasPM As Double
150     Public totTransCaliGasPM As Double
151     Public totTransConvDiesPM As Double
152     Public totTransLowDiesPM As Double
153     Public totTransConvGasCO2 As Double
154     Public totTransFedGasCO2 As Double
155     Public totTransCaliGasCO2 As Double
156     Public totTransConvDiesCO2 As Double
157     Public totTransLowDiesCO2 As Double
158     Public totTransConvGasGHG As Double
159     Public totTransFedGasGHG As Double
160     Public totTransCaliGasGHG As Double
161     Public totTransConvDiesGHG As Double
162     Public totTransLowDiesGHG As Double
163     ' Total Emissions by Stage of Fuel Production and Distribution (grams/million Btu):
        petroleum refining
164     Public totRefConvGasCO As Double
165     Public totRefFedGasCO As Double
166     Public totRefCaliGasCO As Double
167     Public totRefConvDiesCO As Double
168     Public totRefLowDiesCO As Double
169     Public totRefConvGasVOC As Double
170     Public totRefFedGasVOC As Double
171     Public totRefCaliGasVOC As Double
172     Public totRefConvDiesVOC As Double
173     Public totRefLowDiesVOC As Double
174     Public totRefConvGasNOX As Double
175     Public totRefFedGasNOX As Double
176     Public totRefCaliGasNOX As Double
177     Public totRefConvDiesNOX As Double
178     Public totRefLowDiesNOX As Double
179     Public totRefConvGasSOX As Double
180     Public totRefFedGasSOX As Double
181     Public totRefCaliGasSOX As Double
182     Public totRefConvDiesSOX As Double
183     Public totRefLowDiesSOX As Double
184     Public totRefConvGasPM As Double
185     Public totRefFedGasPM As Double
186     Public totRefCaliGasPM As Double
187     Public totRefConvDiesPM As Double
188     Public totRefLowDiesPM As Double
189     Public totRefConvGasCO2 As Double
190     Public totRefFedGasCO2 As Double
191     Public totRefCaliGasCO2 As Double
192     Public totRefConvDiesCO2 As Double
```

```
193     Public totRefLowDiesCO2 As Double
194     Public totRefConvGasGHG As Double
195     Public totRefFedGasGHG As Double
196     Public totRefCaliGasGHG As Double
197     Public totRefConvDiesGHG As Double
198     Public totRefLowDiesGHG As Double
199     ' Total Emissions by Stage of Fuel Production and Distribution (grams/million Btu): fuel ↙
         transportation,
200     ' storage, and distribution of refined gasoline and diesel fuel
201     Public totTSDConvGasCO As Double
202     Public totTSDFedGasCO As Double
203     Public totTSDCaliGasCO As Double
204     Public totTSDConvDiesCO As Double
205     Public totTSDLowDiesCO As Double
206     Public totTSDConvGasVOC As Double
207     Public totTSDFedGasVOC As Double
208     Public totTSDCaliGasVOC As Double
209     Public totTSDConvDiesVOC As Double
210     Public totTSDLowDiesVOC As Double
211     Public totTSDConvGasNOX As Double
212     Public totTSDFedGasNOX As Double
213     Public totTSDCaliGasNOX As Double
214     Public totTSDConvDiesNOX As Double
215     Public totTSDLowDiesNOX As Double
216     Public totTSDConvGasSOX As Double
217     Public totTSDFedGasSOX As Double
218     Public totTSDCaliGasSOX As Double
219     Public totTSDConvDiesSOX As Double
220     Public totTSDLowDiesSOX As Double
221     Public totTSDConvGasPM As Double
222     Public totTSDFedGasPM As Double
223     Public totTSDCaliGasPM As Double
224     Public totTSDConvDiesPM As Double
225     Public totTSDLowDiesPM As Double
226     Public totTSDConvGasCO2 As Double
227     Public totTSDFedGasCO2 As Double
228     Public totTSDCaliGasCO2 As Double
229     Public totTSDConvDiesCO2 As Double
230     Public totTSDLowDiesCO2 As Double
231     Public totTSDConvGasGHG As Double
232     Public totTSDFedGasGHG As Double
233     Public totTSDCaliGasGHG As Double
234     Public totTSDConvDiesGHG As Double
235     Public totTSDLowDiesGHG As Double
236     'Emissions in Urban Areas by Stage of Fuel Production and Distribution (grams/million ↙
        Btu)
237     Public urbExtrConvGasCO As Double
238     Public urbExtrFedGasCO As Double
239     Public urbExtrCaliGasCO As Double
240     Public urbExtrConvDiesCO As Double
241     Public urbExtrLowDiesCO As Double
242     Public urbExtrConvGasVOC As Double
243     Public urbExtrFedGasVOC As Double
244     Public urbExtrCaliGasVOC As Double
245     Public urbExtrConvDiesVOC As Double
246     Public urbExtrLowDiesVOC As Double
247     Public urbExtrConvGasNOX As Double
248     Public urbExtrFedGasNOX As Double
249     Public urbExtrCaliGasNOX As Double
250     Public urbExtrConvDiesNOX As Double
251     Public urbExtrLowDiesNOX As Double
252     Public urbExtrConvGasSOX As Double
253     Public urbExtrFedGasSOX As Double
254     Public urbExtrCaliGasSOX As Double
255     Public urbExtrConvDiesSOX As Double
256     Public urbExtrLowDiesSOX As Double
257     Public urbExtrConvGasPM As Double
258     Public urbExtrFedGasPM As Double
259     Public urbExtrCaliGasPM As Double
```

```
260     Public urbExtrConvDiesPM As Double
261     Public urbExtrLowDiesPM As Double
262     Public urbExtrConvGasCO2 As Double
263     Public urbExtrFedGasCO2 As Double
264     Public urbExtrCaliGasCO2 As Double
265     Public urbExtrConvDiesCO2 As Double
266     Public urbExtrLowDiesCO2 As Double
267     Public urbExtrConvGasGHG As Double
268     Public urbExtrFedGasGHG As Double
269     Public urbExtrCaliGasGHG As Double
270     Public urbExtrConvDiesGHG As Double
271     Public urbExtrLowDiesGHG As Double
272     Public urbTransConvGasCO As Double
273     Public urbTransFedGasCO As Double
274     Public urbTransCaliGasCO As Double
275     Public urbTransConvDiesCO As Double
276     Public urbTransLowDiesCO As Double
277     Public urbTransConvGasVOC As Double
278     Public urbTransFedGasVOC As Double
279     Public urbTransCaliGasVOC As Double
280     Public urbTransConvDiesVOC As Double
281     Public urbTransLowDiesVOC As Double
282     Public urbTransConvGasNOX As Double
283     Public urbTransFedGasNOX As Double
284     Public urbTransCaliGasNOX As Double
285     Public urbTransConvDiesNOX As Double
286     Public urbTransLowDiesNOX As Double
287     Public urbTransConvGasSOX As Double
288     Public urbTransFedGasSOX As Double
289     Public urbTransCaliGasSOX As Double
290     Public urbTransConvDiesSOX As Double
291     Public urbTransLowDiesSOX As Double
292     Public urbTransConvGasPM As Double
293     Public urbTransFedGasPM As Double
294     Public urbTransCaliGasPM As Double
295     Public urbTransConvDiesPM As Double
296     Public urbTransLowDiesPM As Double
297     Public urbTransConvGasCO2 As Double
298     Public urbTransFedGasCO2 As Double
299     Public urbTransCaliGasCO2 As Double
300     Public urbTransConvDiesCO2 As Double
301     Public urbTransLowDiesCO2 As Double
302     Public urbTransConvGasGHG As Double
303     Public urbTransFedGasGHG As Double
304     Public urbTransCaliGasGHG As Double
305     Public urbTransConvDiesGHG As Double
306     Public urbTransLowDiesGHG As Double
307     Public urbRefConvGasCO As Double
308     Public urbRefFedGasCO As Double
309     Public urbRefCaliGasCO As Double
310     Public urbRefConvDiesCO As Double
311     Public urbRefLowDiesCO As Double
312     Public urbRefConvGasVOC As Double
313     Public urbRefFedGasVOC As Double
314     Public urbRefCaliGasVOC As Double
315     Public urbRefConvDiesVOC As Double
316     Public urbRefLowDiesVOC As Double
317     Public urbRefConvGasNOX As Double
318     Public urbRefFedGasNOX As Double
319     Public urbRefCaliGasNOX As Double
320     Public urbRefConvDiesNOX As Double
321     Public urbRefLowDiesNOX As Double
322     Public urbRefConvGasSOX As Double
323     Public urbRefFedGasSOX As Double
324     Public urbRefCaliGasSOX As Double
325     Public urbRefConvDiesSOX As Double
326     Public urbRefLowDiesSOX As Double
327     Public urbRefConvGasPM As Double
328     Public urbRefFedGasPM As Double
```

```
329     Public urbRefCaliGasPM As Double
330     Public urbRefConvDiesPM As Double
331     Public urbRefLowDiesPM As Double
332     Public urbRefConvGasCO2 As Double
333     Public urbRefFedGasCO2 As Double
334     Public urbRefCaliGasCO2 As Double
335     Public urbRefConvDiesCO2 As Double
336     Public urbRefLowDiesCO2 As Double
337     Public urbRefConvGasGHG As Double
338     Public urbRefFedGasGHG As Double
339     Public urbRefCaliGasGHG As Double
340     Public urbRefConvDiesGHG As Double
341     Public urbRefLowDiesGHG As Double
342     Public urbTSDConvGasCO As Double
343     Public urbTSDFedGasCO As Double
344     Public urbTSDCaliGasCO As Double
345     Public urbTSDConvDiesCO As Double
346     Public urbTSDLowDiesCO As Double
347     Public urbTSDConvGasVOC As Double
348     Public urbTSDFedGasVOC As Double
349     Public urbTSDCaliGasVOC As Double
350     Public urbTSDConvDiesVOC As Double
351     Public urbTSDLowDiesVOC As Double
352     Public urbTSDConvGasNOX As Double
353     Public urbTSDFedGasNOX As Double
354     Public urbTSDCaliGasNOX As Double
355     Public urbTSDConvDiesNOX As Double
356     Public urbTSDLowDiesNOX As Double
357     Public urbTSDConvGasSOX As Double
358     Public urbTSDFedGasSOX As Double
359     Public urbTSDCaliGasSOX As Double
360     Public urbTSDConvDiesSOX As Double
361     Public urbTSDLowDiesSOX As Double
362     Public urbTSDConvGasPM As Double
363     Public urbTSDFedGasPM As Double
364     Public urbTSDCaliGasPM As Double
365     Public urbTSDConvDiesPM As Double
366     Public urbTSDLowDiesPM As Double
367     Public urbTSDConvGasCO2 As Double
368     Public urbTSDFedGasCO2 As Double
369     Public urbTSDCaliGasCO2 As Double
370     Public urbTSDConvDiesCO2 As Double
371     Public urbTSDLowDiesCO2 As Double
372     Public urbTSDConvGasGHG As Double
373     Public urbTSDFedGasGHG As Double
374     Public urbTSDCaliGasGHG As Double
375     Public urbTSDConvDiesGHG As Double
376     Public urbTSDLowDiesGHG As Double
377
378     ' set up fleet parameters vars
379     Public percentSalesPreviousYear As Double      ' % of Calendar Year(t-1) Sales that are
        Model Year(t) Vehicles
380     Public percentSalesCurrentYear As Double       ' % of Calendar Year(t) Sales that are
        Model Year(t) Vehicles
381     Public percentSalesFollowingYear As Double     ' % of Calendar Year(t+1) Sales that are
        Model Year(t) Vehicles
382     Public percentLDGT1 As Double                  ' % of Light Truck Class LDT1 Consisting
        of MOBILE6 Class LDGT1
383     Public percentLDGT2 As Double                  ' % of Light Truck Class LDT1 Consisting
        of MOBILE6 Class LDGT2
384     Public percentLDDT12 As Double                 ' % of Light Truck Class LDT1 Consisting
        of MOBILE6 Class LDDT12
385     Public percentLDGT3 As Double                  ' % of Light Truck Class LDT2 Consisting
        of MOBILE6 Class LDGT3
386     Public percentLDGT4 As Double                  ' % of Light Truck Class LDT2 Consisting
        of MOBILE6 Class LDGT4
387     Public percentLDDT34 As Double                 ' % of Light Truck Class LDT2 Consisting
        of MOBILE6 Class LDDT34
388     Public gapTestOnRoad As Double                 ' "Gap" between Test and On-Road MPG
```

```
389     Public fuelTankLT6000 As Double                 ' average fuel tank capacity for light ↙
        trucks under 6000 lbs GVW
390     Public fuelTankLT60008500 As Double             ' average fuel tank capacity for light ↙
        trucks 6000 - 8500 lbs GVW
391     Public fuelTankLT8500 As Double                 ' average fuel tank capacity for light ↙
        trucks over 8500 lbs GVW
392     Public fuelTankCars As Double                   ' average fuel tank capacity for cars
393
394     ' set up economic value vars
395     Public reboundEffect As Double                  ' rebound effect (VMT elasticity wrt fuel ↙
         cost per mile)
396     Public discountRate As Double                   ' discount rate applied to future effects
397     ' economic costs of oil imports ($/gal)
398     Public monopsonyCost As Double                  ' monopsony component
399     Public priceShockCost As Double                 ' price shock component
400     Public militarySecurityCost As Double           ' military security component
401     Public totalEconomicCost As Double              ' total economic costs
402     Public totalEconomicCostBBL As Double           ' total economic costs ($/BBL)
403     ' external costs from additional vehicle use due to "rebound effect" ($/vehicle mile)
404     Public congestionCost As Double                 ' congestion
405     Public accidentCost As Double                   ' accidents
406     Public noiseCost As Double                      ' noise
407     Public totalExternalCost As Double              ' total
408     ' emission damage costs
409     Public damageCostCO As Double                    ' carbon monoxide ($/ton)
410     Public damageCostVOC As Double                   ' volatile organic compounds ($/ton)
411     Public damageCostNOX As Double                   ' nitrogen oxides ($/ton)
412     Public damageCostPM As Double                    ' particulate matter ($/ton)
413     Public damageCostSO2 As Double                   ' sulfur dioxide ($/ton)
414     Public damageCostC As Double                     ' carbon ($/metric ton)
415     Public damageCostCO2 As Double                   ' carbon dioxide ($/metric ton)
416     ' value of travel time
417     Public travelValuePerson As Double               ' value per person ($/hour)
418     Public averageVehicleOccupancy As Double         ' average vehicle occupancy (persons)
419     Public travelValueVehicle As Double              ' value per vehicle ($/hour)
420
421     'set up forecast data vars ... into arrays ... from 2000 - 2025
422     ' new auto sales (thousands):
423     ' this value is not actually used in this program
424     ' value was used to adjust sales values provided by mfgs
425     ' actual sales value
426     Public gasSalesAuto(0, 25) As Double             ' gas
427     Public diesSalesAuto(0, 25) As Double            ' diesel
428     Public ethSalesAuto(0, 25) As Double             ' ethanol
429     Public totSalesAuto(0, 25) As Double             ' total
430     ' new light truck sales (000s):
431     Public gasSalesLT(0, 25) As Double               ' gas
432     Public diesSalesLT(0, 25) As Double              ' diesel
433     Public ethSalesLT(0, 25) As Double               ' ethanol
434     Public totSalesLT(0, 25) As Double               ' total
435     ' EPA fuel economy rating (MPG):
436     Public ratingEPAAuto(0, 25) As Double            ' auto
437     Public ratingEPALT(0, 25) As Double              ' light truck
438     ' ratio: on-road to epa test MPG:
439     Public ratioMPGAuto(0, 25) As Double             ' auto
440     Public ratioMPGLT(0, 25) As Double               ' light truck
441     ' retail fuel prices per gallon (2001 $):
442     Public fuelPriceGas(25) As Double              ' gas
443     Public fuelPriceDiesel(25) As Double           ' diesel
444     ' fuel taxes per gallon:
445     Public fuelTaxGasFed(25) As Double             ' gas federal
446     Public fuelTaxGasState(25) As Double           ' gas state
447     Public fuelTaxGasTot(25) As Double             ' gas total
448     Public fuelTaxDiesFed(25) As Double            ' diesel federal
449     Public fuelTaxDiesState(25) As Double          ' diesel state
450     Public fuelTaxDiesTot(25) As Double            ' diesel total
451     ' pre-tax fuel price per gallon (2001 $)
452     Public fuelPretaxGas(25) As Double             ' gas
453     Public fuelPretaxDies(25) As Double            ' diesel
```

```
454
455
456

457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495

496
497
498
499
500
501
502
503
504
505
506
507

508
509
510
511
512
513
514
515
516
517     'set up emissions values from mobile6
518     Public coGasLDGV(,) As Double
519     Public coGasLDGT1(,) As Double
```

```
520     Public coGasLDGT2(,) As Double
521     Public coGasHDGV2b(,) As Double
522
523     Public coGasRFGLDGV(,) As Double
524     Public coGasRFGLDGT1(,) As Double
525     Public coGasRFGLDGT2(,) As Double
526     Public coGasRFGHDGV2b(,) As Double
527
528     Public coDiesLDDV(,) As Double
529     Public coDiesLDDT1(,) As Double
530     Public coDiesLDDT2(,) As Double
531     Public coDiesHDDV2b(,) As Double
532
533     Public vocGasLDGV(,) As Double
534     Public vocGasLDGT1(,) As Double
535     Public vocGasLDGT2(,) As Double
536     Public vocGasHDGV2b(,) As Double
537
538     Public vocGasRFGLDGV(,) As Double
539     Public vocGasRFGLDGT1(,) As Double
540     Public vocGasRFGLDGT2(,) As Double
541     Public vocGasRFGHDGV2b(,) As Double
542
543     Public vocDiesLDDV(,) As Double
544     Public vocDiesLDDT1(,) As Double
545     Public vocDiesLDDT2(,) As Double
546     Public vocDiesHDDV2b(,) As Double
547
548     Public noxGasLDGV(,) As Double
549     Public noxGasLDGT1(,) As Double
550     Public noxGasLDGT2(,) As Double
551     Public noxGasHDGV2b(,) As Double
552
553     Public noxGasRFGLDGV(,) As Double
554     Public noxGasRFGLDGT1(,) As Double
555     Public noxGasRFGLDGT2(,) As Double
556     Public noxGasRFGHDGV2b(,) As Double
557
558     Public noxDiesLDDV(,) As Double
559     Public noxDiesLDDT1(,) As Double
560     Public noxDiesLDDT2(,) As Double
561     Public noxDiesHDDV2b(,) As Double
562
563     Public pmGasLDGV(,) As Double
564     Public pmGasLDGT1(,) As Double
565     Public pmGasLDGT2(,) As Double
566     Public pmGasHDGV2b(,) As Double
567
568     Public pmGasRFGLDGV(,) As Double
569     Public pmGasRFGLDGT1(,) As Double
570     Public pmGasRFGLDGT2(,) As Double
571     Public pmGasRFGHDGV2b(,) As Double
572
573     Public pmDiesLDDV(,) As Double
574     Public pmDiesLDDT1(,) As Double
575     Public pmDiesLDDT2(,) As Double
576     Public pmDiesHDDV2b(,) As Double
577
578     Public so2GasLDGV(,) As Double
579     Public so2GasLDGT1(,) As Double
580     Public so2GasLDGT2(,) As Double
581     Public so2GasHDGV2b(,) As Double
582
583     Public so2GasRFGLDGV(,) As Double
584     Public so2GasRFGLDGT1(,) As Double
585     Public so2GasRFGLDGT2(,) As Double
586     Public so2GasRFGHDGV2b(,) As Double
587
588     Public so2DiesLDDV(,) As Double
```

```
589     Public so2DiesLDDT1(,) As Double
590     Public so2DiesLDDT2(,) As Double
591     Public so2DiesHDDV2b(,) As Double
592
593     Public RatesMinMY As Integer
594     Public RatesMaxMY As Integer
595     Public RatesMinCY As Integer
596     Public RatesMaxCY As Integer
597
598 #End Region
599
600
601     '
602     ' copy methods
603     '
604     Public Sub Copy(ByRef Dst As ParametersClass)
605         Copy(Dst, Me)
606     End Sub
607
608     Public Shared Sub Copy(ByRef Dst As ParametersClass, ByRef Src As ParametersClass)
609         Dst = New ParametersClass
610
611         CopyComplianceModelData(Dst, Src)
612         CopyEffectsModelData(Dst, Src)
613     End Sub
614
615     Private Shared Sub CopyComplianceModelData(ByRef Dst As ParametersClass, ByRef Src As
        ParametersClass)
616         Dim i, j As Integer
617
618         Dst.Payback = Src.Payback
619         Dst.FE_Shortfall = Src.FE_Shortfall
620         Dst.K_f = Src.K_f
621         Dst.kCurbWeightDelta = Src.kCurbWeightDelta
622
623         Dst.wtpfnMY = Src.wtpfnMY
624         Dst.nMfg = Src.nMfg
625         Dst.nMktSegments = Src.nMktSegments
626         Dst.wtpfMinYear = Src.wtpfMinYear
627         Dst.wtpfMaxYear = Src.wtpfMaxYear
628
629         ReDim Dst.wtpfMY(Dst.wtpfnMY - 1)
630         ReDim Dst.MfgIDs(Dst.nMfg - 1)
631         ReDim Dst.CostAllocationStrategy(Dst.nMfg - 1)
632         ReDim Dst.WillingToPayFinesMfgList(Dst.nMfg - 1)
633         ReDim Dst.WillingToPayFines(CAFE.GMYI(Dst.wtpfMaxYear, False), Dst.nMfg - 1)
634         ReDim Dst.MY02LTWeight(Src.MY02LTWeight.Length - 1)
635         ReDim Dst.Kmy(Src.Kmy.Length - 1)
636         If CAFE.NMNLIncluded Then
637             ReDim Dst.NMNLCoeff(Src.NMNLCoeff.Length - 1)
638         End If
639
640         For i = 0 To Dst.wtpfnMY - 1
641             Dst.wtpfMY(i) = Src.wtpfMY(i)
642         Next i
643
644         For i = 0 To Dst.nMfg - 1
645             Dst.MfgIDs(i) = Src.MfgIDs(i)
646             Dst.CostAllocationStrategy(i) = Src.CostAllocationStrategy(i)
647             Dst.WillingToPayFinesMfgList(i) = Src.WillingToPayFinesMfgList(i)
648
649             For j = 0 To CAFE.GMYI(Dst.wtpfMaxYear, False)
650                 Dst.WillingToPayFines(j, i) = Src.WillingToPayFines(j, i)
651             Next j
652         Next i
653
654         If CAFE.NMNLIncluded Then
655             For i = 0 To Src.NMNLCoeff.Length - 1
656                 Dst.NMNLCoeff(i) = Src.NMNLCoeff(i)
```

```
657             Next i
658         End If
659
660         For i = 0 To Src.MY02LTWeight.Length - 1
661             Dst.MY02LTWeight(i) = Src.MY02LTWeight(i)
662         Next i
663
664         For i = 0 To Src.Kmy.Length - 1
665             Src.Kmy(i).Copy(Dst.Kmy(i))
666         Next i
667     End Sub
668
669     Private Shared Sub CopyEffectsModelData(ByRef Dst As ParametersClass, ByRef Src As
        ParametersClass)
670         Dim i, j As Integer
671
672         j = 0
673         For i = 0 To 25
674             Dst.milesAuto(i, j) = Src.milesAuto(i, j)
675             Dst.milesPickup(i, j) = Src.milesPickup(i, j)
676             Dst.milesVan(i, j) = Src.milesVan(i, j)
677             Dst.milesSuv(i, j) = Src.milesSuv(i, j)
678             Dst.milesLTDefault(i, j) = Src.milesLTDefault(i, j)
679
680             ''Dst.milesLDGV(i, j) = Src.milesLDGV(i, j)
681             ''Dst.milesLDDV(i, j) = Src.milesLDDV(i, j)
682             ''Dst.milesLDGT1(i, j) = Src.milesLDGT1(i, j)
683             ''Dst.milesLDGT2(i, j) = Src.milesLDGT2(i, j)
684             ''Dst.milesLDDV1(i, j) = Src.milesLDDV1(i, j)
685             ''Dst.milesLDDV2(i, j) = Src.milesLDDV2(i, j)
686             ''Dst.milesHDGV2b(i, j) = Src.milesHDGV2b(i, j)
687             ''Dst.milesHDDV2b(i, j) = Src.milesHDDV2b(i, j)
688
689             Dst.survivalLDGV(i, j) = Src.survivalLDGV(i, j)
690             Dst.survivalLDDV(i, j) = Src.survivalLDDV(i, j)
691             Dst.survivalLDGT1(i, j) = Src.survivalLDGT1(i, j)
692             Dst.survivalLDGT2(i, j) = Src.survivalLDGT2(i, j)
693             Dst.survivalLDDV1(i, j) = Src.survivalLDDV1(i, j)
694             Dst.survivalLDDV2(i, j) = Src.survivalLDDV2(i, j)
695             Dst.survivalHDGV2b(i, j) = Src.survivalHDGV2b(i, j)
696             Dst.survivalHDDV2b(i, j) = Src.survivalHDDV2b(i, j)
697         Next i
698
699         Dst.convGasEngDens = Src.convGasEngDens
700         Dst.fedGasEngDens = Src.fedGasEngDens
701         Dst.caliGasEngDens = Src.caliGasEngDens
702         Dst.convDiesEngDens = Src.convDiesEngDens
703         Dst.lowDiesEngDens = Src.lowDiesEngDens
704         Dst.convGasMassDens = Src.convGasMassDens
705         Dst.fedGasMassDens = Src.fedGasMassDens
706         Dst.caliGasMassDens = Src.caliGasMassDens
707         Dst.convDiesMassDens = Src.convDiesMassDens
708         Dst.lowDiesMassDens = Src.lowDiesMassDens
709         Dst.convGasCarb = Src.convGasCarb
710         Dst.fedGasCarb = Src.fedGasCarb
711         Dst.caliGasCarb = Src.caliGasCarb
712         Dst.convDiesCarb = Src.convDiesCarb
713         Dst.lowDiesCarb = Src.lowDiesCarb
714         Dst.convGasSulf = Src.convGasSulf
715         Dst.fedGasSulf = Src.fedGasSulf
716         Dst.caliGasSulf = Src.caliGasSulf
717         Dst.convDiesSulf = Src.convDiesSulf
718         Dst.lowDiesSulf = Src.lowDiesSulf
719         Dst.gasShareImp = Src.gasShareImp
720         Dst.gasShareDomRef = Src.gasShareDomRef
721         Dst.gasShareDomRefDomCrude = Src.gasShareDomRefDomCrude
722         Dst.gasShareDomRefImpCrude = Src.gasShareDomRefImpCrude
723         Dst.gasLowImp = Src.gasLowImp
724         Dst.gasLowDomRef = Src.gasLowDomRef
```

```
725         Dst.gasLowDomRefDomCrude = Src.gasLowDomRefDomCrude
726         Dst.gasLowDomRefImpCrude = Src.gasLowDomRefImpCrude
727         Dst.diesShareImp = Src.diesShareImp
728         Dst.diesShareDomRef = Src.diesShareDomRef
729         Dst.diesShareDomRefDomCrude = Src.diesShareDomRefDomCrude
730         Dst.diesShareDomRefImpCrude = Src.diesShareDomRefImpCrude
731         Dst.diesLowImp = Src.diesLowImp
732         Dst.diesLowDomRef = Src.diesLowDomRef
733         Dst.diesLowDomRefDomCrude = Src.diesLowDomRefDomCrude
734         Dst.diesLowDomRefImpCrude = Src.diesLowDomRefImpCrude
735         Dst.convGasMix = Src.convGasMix
736         Dst.fedGasMix = Src.fedGasMix
737         Dst.fedNorthGasMix = Src.fedNorthGasMix
738         Dst.fedSouthGasMix = Src.fedSouthGasMix
739         Dst.caliGasMix = Src.caliGasMix
740         Dst.convDiesMix = Src.convDiesMix
741         Dst.lowDiesMix = Src.lowDiesMix
742         Dst.totExtrConvGasCO = Src.totExtrConvGasCO
743         Dst.totExtrFedGasCO = Src.totExtrFedGasCO
744         Dst.totExtrCaliGasCO = Src.totExtrCaliGasCO
745         Dst.totExtrConvDiesCO = Src.totExtrConvDiesCO
746         Dst.totExtrLowDiesCO = Src.totExtrLowDiesCO
747         Dst.totExtrConvGasVOC = Src.totExtrConvGasVOC
748         Dst.totExtrFedGasVOC = Src.totExtrFedGasVOC
749         Dst.totExtrCaliGasVOC = Src.totExtrCaliGasVOC
750         Dst.totExtrConvDiesVOC = Src.totExtrConvDiesVOC
751         Dst.totExtrLowDiesVOC = Src.totExtrLowDiesVOC
752         Dst.totExtrConvGasNOX = Src.totExtrConvGasNOX
753         Dst.totExtrFedGasNOX = Src.totExtrFedGasNOX
754         Dst.totExtrCaliGasNOX = Src.totExtrCaliGasNOX
755         Dst.totExtrConvDiesNOX = Src.totExtrConvDiesNOX
756         Dst.totExtrLowDiesNOX = Src.totExtrLowDiesNOX
757         Dst.totExtrConvGasSOX = Src.totExtrConvGasSOX
758         Dst.totExtrFedGasSOX = Src.totExtrFedGasSOX
759         Dst.totExtrCaliGasSOX = Src.totExtrCaliGasSOX
760         Dst.totExtrConvDiesSOX = Src.totExtrConvDiesSOX
761         Dst.totExtrLowDiesSOX = Src.totExtrLowDiesSOX
762         Dst.totExtrConvGasPM = Src.totExtrConvGasPM
763         Dst.totExtrFedGasPM = Src.totExtrFedGasPM
764         Dst.totExtrCaliGasPM = Src.totExtrCaliGasPM
765         Dst.totExtrConvDiesPM = Src.totExtrConvDiesPM
766         Dst.totExtrLowDiesPM = Src.totExtrLowDiesPM
767         Dst.totExtrConvGasCO2 = Src.totExtrConvGasCO2
768         Dst.totExtrFedGasCO2 = Src.totExtrFedGasCO2
769         Dst.totExtrCaliGasCO2 = Src.totExtrCaliGasCO2
770         Dst.totExtrConvDiesCO2 = Src.totExtrConvDiesCO2
771         Dst.totExtrLowDiesCO2 = Src.totExtrLowDiesCO2
772         Dst.totExtrConvGasGHG = Src.totExtrConvGasGHG
773         Dst.totExtrFedGasGHG = Src.totExtrFedGasGHG
774         Dst.totExtrCaliGasGHG = Src.totExtrCaliGasGHG
775         Dst.totExtrConvDiesGHG = Src.totExtrConvDiesGHG
776         Dst.totExtrLowDiesGHG = Src.totExtrLowDiesGHG
777         Dst.totTransConvGasCO = Src.totTransConvGasCO
778         Dst.totTransFedGasCO = Src.totTransFedGasCO
779         Dst.totTransCaliGasCO = Src.totTransCaliGasCO
780         Dst.totTransConvDiesCO = Src.totTransConvDiesCO
781         Dst.totTransLowDiesCO = Src.totTransLowDiesCO
782         Dst.totTransConvGasVOC = Src.totTransConvGasVOC
783         Dst.totTransFedGasVOC = Src.totTransFedGasVOC
784         Dst.totTransCaliGasVOC = Src.totTransCaliGasVOC
785         Dst.totTransConvDiesVOC = Src.totTransConvDiesVOC
786         Dst.totTransLowDiesVOC = Src.totTransLowDiesVOC
787         Dst.totTransConvGasNOX = Src.totTransConvGasNOX
788         Dst.totTransFedGasNOX = Src.totTransFedGasNOX
789         Dst.totTransCaliGasNOX = Src.totTransCaliGasNOX
790         Dst.totTransConvDiesNOX = Src.totTransConvDiesNOX
791         Dst.totTransLowDiesNOX = Src.totTransLowDiesNOX
792         Dst.totTransConvGasSOX = Src.totTransConvGasSOX
793         Dst.totTransFedGasSOX = Src.totTransFedGasSOX
```

```
794         Dst.totTransCaliGasSOX = Src.totTransCaliGasSOX
795         Dst.totTransConvDiesSOX = Src.totTransConvDiesSOX
796         Dst.totTransLowDiesSOX = Src.totTransLowDiesSOX
797         Dst.totTransConvGasPM = Src.totTransConvGasPM
798         Dst.totTransFedGasPM = Src.totTransFedGasPM
799         Dst.totTransCaliGasPM = Src.totTransCaliGasPM
800         Dst.totTransConvDiesPM = Src.totTransConvDiesPM
801         Dst.totTransLowDiesPM = Src.totTransLowDiesPM
802         Dst.totTransConvGasCO2 = Src.totTransConvGasCO2
803         Dst.totTransFedGasCO2 = Src.totTransFedGasCO2
804         Dst.totTransCaliGasCO2 = Src.totTransCaliGasCO2
805         Dst.totTransConvDiesCO2 = Src.totTransConvDiesCO2
806         Dst.totTransLowDiesCO2 = Src.totTransLowDiesCO2
807         Dst.totTransConvGasGHG = Src.totTransConvGasGHG
808         Dst.totTransFedGasGHG = Src.totTransFedGasGHG
809         Dst.totTransCaliGasGHG = Src.totTransCaliGasGHG
810         Dst.totTransConvDiesGHG = Src.totTransConvDiesGHG
811         Dst.totTransLowDiesGHG = Src.totTransLowDiesGHG
812         Dst.totRefConvGasCO = Src.totRefConvGasCO
813         Dst.totRefFedGasCO = Src.totRefFedGasCO
814         Dst.totRefCaliGasCO = Src.totRefCaliGasCO
815         Dst.totRefConvDiesCO = Src.totRefConvDiesCO
816         Dst.totRefLowDiesCO = Src.totRefLowDiesCO
817         Dst.totRefConvGasVOC = Src.totRefConvGasVOC
818         Dst.totRefFedGasVOC = Src.totRefFedGasVOC
819         Dst.totRefCaliGasVOC = Src.totRefCaliGasVOC
820         Dst.totRefConvDiesVOC = Src.totRefConvDiesVOC
821         Dst.totRefLowDiesVOC = Src.totRefLowDiesVOC
822         Dst.totRefConvGasNOX = Src.totRefConvGasNOX
823         Dst.totRefFedGasNOX = Src.totRefFedGasNOX
824         Dst.totRefCaliGasNOX = Src.totRefCaliGasNOX
825         Dst.totRefConvDiesNOX = Src.totRefConvDiesNOX
826         Dst.totRefLowDiesNOX = Src.totRefLowDiesNOX
827         Dst.totRefConvGasSOX = Src.totRefConvGasSOX
828         Dst.totRefFedGasSOX = Src.totRefFedGasSOX
829         Dst.totRefCaliGasSOX = Src.totRefCaliGasSOX
830         Dst.totRefConvDiesSOX = Src.totRefConvDiesSOX
831         Dst.totRefLowDiesSOX = Src.totRefLowDiesSOX
832         Dst.totRefConvGasPM = Src.totRefConvGasPM
833         Dst.totRefFedGasPM = Src.totRefFedGasPM
834         Dst.totRefCaliGasPM = Src.totRefCaliGasPM
835         Dst.totRefConvDiesPM = Src.totRefConvDiesPM
836         Dst.totRefLowDiesPM = Src.totRefLowDiesPM
837         Dst.totRefConvGasCO2 = Src.totRefConvGasCO2
838         Dst.totRefFedGasCO2 = Src.totRefFedGasCO2
839         Dst.totRefCaliGasCO2 = Src.totRefCaliGasCO2
840         Dst.totRefConvDiesCO2 = Src.totRefConvDiesCO2
841         Dst.totRefLowDiesCO2 = Src.totRefLowDiesCO2
842         Dst.totRefConvGasGHG = Src.totRefConvGasGHG
843         Dst.totRefFedGasGHG = Src.totRefFedGasGHG
844         Dst.totRefCaliGasGHG = Src.totRefCaliGasGHG
845         Dst.totRefConvDiesGHG = Src.totRefConvDiesGHG
846         Dst.totRefLowDiesGHG = Src.totRefLowDiesGHG
847         Dst.totTSDConvGasCO = Src.totTSDConvGasCO
848         Dst.totTSDFedGasCO = Src.totTSDFedGasCO
849         Dst.totTSDCaliGasCO = Src.totTSDCaliGasCO
850         Dst.totTSDConvDiesCO = Src.totTSDConvDiesCO
851         Dst.totTSDLowDiesCO = Src.totTSDLowDiesCO
852         Dst.totTSDConvGasVOC = Src.totTSDConvGasVOC
853         Dst.totTSDFedGasVOC = Src.totTSDFedGasVOC
854         Dst.totTSDCaliGasVOC = Src.totTSDCaliGasVOC
855         Dst.totTSDConvDiesVOC = Src.totTSDConvDiesVOC
856         Dst.totTSDLowDiesVOC = Src.totTSDLowDiesVOC
857         Dst.totTSDConvGasNOX = Src.totTSDConvGasNOX
858         Dst.totTSDFedGasNOX = Src.totTSDFedGasNOX
859         Dst.totTSDCaliGasNOX = Src.totTSDCaliGasNOX
860         Dst.totTSDConvDiesNOX = Src.totTSDConvDiesNOX
861         Dst.totTSDLowDiesNOX = Src.totTSDLowDiesNOX
862         Dst.totTSDConvGasSOX = Src.totTSDConvGasSOX
```

```
863         Dst.totTSDFedGasSOX = Src.totTSDFedGasSOX
864         Dst.totTSDCaliGasSOX = Src.totTSDCaliGasSOX
865         Dst.totTSDConvDiesSOX = Src.totTSDConvDiesSOX
866         Dst.totTSDLowDiesSOX = Src.totTSDLowDiesSOX
867         Dst.totTSDConvGasPM = Src.totTSDConvGasPM
868         Dst.totTSDFedGasPM = Src.totTSDFedGasPM
869         Dst.totTSDCaliGasPM = Src.totTSDCaliGasPM
870         Dst.totTSDConvDiesPM = Src.totTSDConvDiesPM
871         Dst.totTSDLowDiesPM = Src.totTSDLowDiesPM
872         Dst.totTSDConvGasCO2 = Src.totTSDConvGasCO2
873         Dst.totTSDFedGasCO2 = Src.totTSDFedGasCO2
874         Dst.totTSDCaliGasCO2 = Src.totTSDCaliGasCO2
875         Dst.totTSDConvDiesCO2 = Src.totTSDConvDiesCO2
876         Dst.totTSDLowDiesCO2 = Src.totTSDLowDiesCO2
877         Dst.totTSDConvGasGHG = Src.totTSDConvGasGHG
878         Dst.totTSDFedGasGHG = Src.totTSDFedGasGHG
879         Dst.totTSDCaliGasGHG = Src.totTSDCaliGasGHG
880         Dst.totTSDConvDiesGHG = Src.totTSDConvDiesGHG
881         Dst.totTSDLowDiesGHG = Src.totTSDLowDiesGHG
882         Dst.urbExtrConvGasCO = Src.urbExtrConvGasCO
883         Dst.urbExtrFedGasCO = Src.urbExtrFedGasCO
884         Dst.urbExtrCaliGasCO = Src.urbExtrCaliGasCO
885         Dst.urbExtrConvDiesCO = Src.urbExtrConvDiesCO
886         Dst.urbExtrLowDiesCO = Src.urbExtrLowDiesCO
887         Dst.urbExtrConvGasVOC = Src.urbExtrConvGasVOC
888         Dst.urbExtrFedGasVOC = Src.urbExtrFedGasVOC
889         Dst.urbExtrCaliGasVOC = Src.urbExtrCaliGasVOC
890         Dst.urbExtrConvDiesVOC = Src.urbExtrConvDiesVOC
891         Dst.urbExtrLowDiesVOC = Src.urbExtrLowDiesVOC
892         Dst.urbExtrConvGasNOX = Src.urbExtrConvGasNOX
893         Dst.urbExtrFedGasNOX = Src.urbExtrFedGasNOX
894         Dst.urbExtrCaliGasNOX = Src.urbExtrCaliGasNOX
895         Dst.urbExtrConvDiesNOX = Src.urbExtrConvDiesNOX
896         Dst.urbExtrLowDiesNOX = Src.urbExtrLowDiesNOX
897         Dst.urbExtrConvGasSOX = Src.urbExtrConvGasSOX
898         Dst.urbExtrFedGasSOX = Src.urbExtrFedGasSOX
899         Dst.urbExtrCaliGasSOX = Src.urbExtrCaliGasSOX
900         Dst.urbExtrConvDiesSOX = Src.urbExtrConvDiesSOX
901         Dst.urbExtrLowDiesSOX = Src.urbExtrLowDiesSOX
902         Dst.urbExtrConvGasPM = Src.urbExtrConvGasPM
903         Dst.urbExtrFedGasPM = Src.urbExtrFedGasPM
904         Dst.urbExtrCaliGasPM = Src.urbExtrCaliGasPM
905         Dst.urbExtrConvDiesPM = Src.urbExtrConvDiesPM
906         Dst.urbExtrLowDiesPM = Src.urbExtrLowDiesPM
907         Dst.urbExtrConvGasCO2 = Src.urbExtrConvGasCO2
908         Dst.urbExtrFedGasCO2 = Src.urbExtrFedGasCO2
909         Dst.urbExtrCaliGasCO2 = Src.urbExtrCaliGasCO2
910         Dst.urbExtrConvDiesCO2 = Src.urbExtrConvDiesCO2
911         Dst.urbExtrLowDiesCO2 = Src.urbExtrLowDiesCO2
912         Dst.urbExtrConvGasGHG = Src.urbExtrConvGasGHG
913         Dst.urbExtrFedGasGHG = Src.urbExtrFedGasGHG
914         Dst.urbExtrCaliGasGHG = Src.urbExtrCaliGasGHG
915         Dst.urbExtrConvDiesGHG = Src.urbExtrConvDiesGHG
916         Dst.urbExtrLowDiesGHG = Src.urbExtrLowDiesGHG
917         Dst.urbTransConvGasCO = Src.urbTransConvGasCO
918         Dst.urbTransFedGasCO = Src.urbTransFedGasCO
919         Dst.urbTransCaliGasCO = Src.urbTransCaliGasCO
920         Dst.urbTransConvDiesCO = Src.urbTransConvDiesCO
921         Dst.urbTransLowDiesCO = Src.urbTransLowDiesCO
922         Dst.urbTransConvGasVOC = Src.urbTransConvGasVOC
923         Dst.urbTransFedGasVOC = Src.urbTransFedGasVOC
924         Dst.urbTransCaliGasVOC = Src.urbTransCaliGasVOC
925         Dst.urbTransConvDiesVOC = Src.urbTransConvDiesVOC
926         Dst.urbTransLowDiesVOC = Src.urbTransLowDiesVOC
927         Dst.urbTransConvGasNOX = Src.urbTransConvGasNOX
928         Dst.urbTransFedGasNOX = Src.urbTransFedGasNOX
929         Dst.urbTransCaliGasNOX = Src.urbTransCaliGasNOX
930         Dst.urbTransConvDiesNOX = Src.urbTransConvDiesNOX
931         Dst.urbTransLowDiesNOX = Src.urbTransLowDiesNOX
```

```
        Dst.urbTransConvGasSOX = Src.urbTransConvGasSOX
        Dst.urbTransFedGasSOX = Src.urbTransFedGasSOX
        Dst.urbTransCaliGasSOX = Src.urbTransCaliGasSOX
        Dst.urbTransConvDiesSOX = Src.urbTransConvDiesSOX
        Dst.urbTransLowDiesSOX = Src.urbTransLowDiesSOX
        Dst.urbTransConvGasPM = Src.urbTransConvGasPM
        Dst.urbTransFedGasPM = Src.urbTransFedGasPM
        Dst.urbTransCaliGasPM = Src.urbTransCaliGasPM
        Dst.urbTransConvDiesPM = Src.urbTransConvDiesPM
        Dst.urbTransLowDiesPM = Src.urbTransLowDiesPM
        Dst.urbTransConvGasCO2 = Src.urbTransConvGasCO2
        Dst.urbTransFedGasCO2 = Src.urbTransFedGasCO2
        Dst.urbTransCaliGasCO2 = Src.urbTransCaliGasCO2
        Dst.urbTransConvDiesCO2 = Src.urbTransConvDiesCO2
        Dst.urbTransLowDiesCO2 = Src.urbTransLowDiesCO2
        Dst.urbTransConvGasGHG = Src.urbTransConvGasGHG
        Dst.urbTransFedGasGHG = Src.urbTransFedGasGHG
        Dst.urbTransCaliGasGHG = Src.urbTransCaliGasGHG
        Dst.urbTransConvDiesGHG = Src.urbTransConvDiesGHG
        Dst.urbTransLowDiesGHG = Src.urbTransLowDiesGHG
        Dst.urbRefConvGasCO = Src.urbRefConvGasCO
        Dst.urbRefFedGasCO = Src.urbRefFedGasCO
        Dst.urbRefCaliGasCO = Src.urbRefCaliGasCO
        Dst.urbRefConvDiesCO = Src.urbRefConvDiesCO
        Dst.urbRefLowDiesCO = Src.urbRefLowDiesCO
        Dst.urbRefConvGasVOC = Src.urbRefConvGasVOC
        Dst.urbRefFedGasVOC = Src.urbRefFedGasVOC
        Dst.urbRefCaliGasVOC = Src.urbRefCaliGasVOC
        Dst.urbRefConvDiesVOC = Src.urbRefConvDiesVOC
        Dst.urbRefLowDiesVOC = Src.urbRefLowDiesVOC
        Dst.urbRefConvGasNOX = Src.urbRefConvGasNOX
        Dst.urbRefFedGasNOX = Src.urbRefFedGasNOX
        Dst.urbRefCaliGasNOX = Src.urbRefCaliGasNOX
        Dst.urbRefConvDiesNOX = Src.urbRefConvDiesNOX
        Dst.urbRefLowDiesNOX = Src.urbRefLowDiesNOX
        Dst.urbRefConvGasSOX = Src.urbRefConvGasSOX
        Dst.urbRefFedGasSOX = Src.urbRefFedGasSOX
        Dst.urbRefCaliGasSOX = Src.urbRefCaliGasSOX
        Dst.urbRefConvDiesSOX = Src.urbRefConvDiesSOX
        Dst.urbRefLowDiesSOX = Src.urbRefLowDiesSOX
        Dst.urbRefConvGasPM = Src.urbRefConvGasPM
        Dst.urbRefFedGasPM = Src.urbRefFedGasPM
        Dst.urbRefCaliGasPM = Src.urbRefCaliGasPM
        Dst.urbRefConvDiesPM = Src.urbRefConvDiesPM
        Dst.urbRefLowDiesPM = Src.urbRefLowDiesPM
        Dst.urbRefConvGasCO2 = Src.urbRefConvGasCO2
        Dst.urbRefFedGasCO2 = Src.urbRefFedGasCO2
        Dst.urbRefCaliGasCO2 = Src.urbRefCaliGasCO2
        Dst.urbRefConvDiesCO2 = Src.urbRefConvDiesCO2
        Dst.urbRefLowDiesCO2 = Src.urbRefLowDiesCO2
        Dst.urbRefConvGasGHG = Src.urbRefConvGasGHG
        Dst.urbRefFedGasGHG = Src.urbRefFedGasGHG
        Dst.urbRefCaliGasGHG = Src.urbRefCaliGasGHG
        Dst.urbRefConvDiesGHG = Src.urbRefConvDiesGHG
        Dst.urbRefLowDiesGHG = Src.urbRefLowDiesGHG
        Dst.urbTSDConvGasCO = Src.urbTSDConvGasCO
        Dst.urbTSDFedGasCO = Src.urbTSDFedGasCO
        Dst.urbTSDCaliGasCO = Src.urbTSDCaliGasCO
        Dst.urbTSDConvDiesCO = Src.urbTSDConvDiesCO
        Dst.urbTSDLowDiesCO = Src.urbTSDLowDiesCO
        Dst.urbTSDConvGasVOC = Src.urbTSDConvGasVOC
        Dst.urbTSDFedGasVOC = Src.urbTSDFedGasVOC
        Dst.urbTSDCaliGasVOC = Src.urbTSDCaliGasVOC
        Dst.urbTSDConvDiesVOC = Src.urbTSDConvDiesVOC
        Dst.urbTSDLowDiesVOC = Src.urbTSDLowDiesVOC
        Dst.urbTSDConvGasNOX = Src.urbTSDConvGasNOX
        Dst.urbTSDFedGasNOX = Src.urbTSDFedGasNOX
        Dst.urbTSDCaliGasNOX = Src.urbTSDCaliGasNOX
        Dst.urbTSDConvDiesNOX = Src.urbTSDConvDiesNOX
```

```
1001        Dst.urbTSDLowDiesNOX = Src.urbTSDLowDiesNOX
1002        Dst.urbTSDConvGasSOX = Src.urbTSDConvGasSOX
1003        Dst.urbTSDFedGasSOX = Src.urbTSDFedGasSOX
1004        Dst.urbTSDCaliGasSOX = Src.urbTSDCaliGasSOX
1005        Dst.urbTSDConvDiesSOX = Src.urbTSDConvDiesSOX
1006        Dst.urbTSDLowDiesSOX = Src.urbTSDLowDiesSOX
1007        Dst.urbTSDConvGasPM = Src.urbTSDConvGasPM
1008        Dst.urbTSDFedGasPM = Src.urbTSDFedGasPM
1009        Dst.urbTSDCaliGasPM = Src.urbTSDCaliGasPM
1010        Dst.urbTSDConvDiesPM = Src.urbTSDConvDiesPM
1011        Dst.urbTSDLowDiesPM = Src.urbTSDLowDiesPM
1012        Dst.urbTSDConvGasCO2 = Src.urbTSDConvGasCO2
1013        Dst.urbTSDFedGasCO2 = Src.urbTSDFedGasCO2
1014        Dst.urbTSDCaliGasCO2 = Src.urbTSDCaliGasCO2
1015        Dst.urbTSDConvDiesCO2 = Src.urbTSDConvDiesCO2
1016        Dst.urbTSDLowDiesCO2 = Src.urbTSDLowDiesCO2
1017        Dst.urbTSDConvGasGHG = Src.urbTSDConvGasGHG
1018        Dst.urbTSDFedGasGHG = Src.urbTSDFedGasGHG
1019        Dst.urbTSDCaliGasGHG = Src.urbTSDCaliGasGHG
1020        Dst.urbTSDConvDiesGHG = Src.urbTSDConvDiesGHG
1021        Dst.urbTSDLowDiesGHG = Src.urbTSDLowDiesGHG
1022        Dst.percentSalesPreviousYear = Src.percentSalesPreviousYear
1023        Dst.percentSalesCurrentYear = Src.percentSalesCurrentYear
1024        Dst.percentSalesFollowingYear = Src.percentSalesFollowingYear
1025        Dst.percentLDGT1 = Src.percentLDGT1
1026        Dst.percentLDGT2 = Src.percentLDGT2
1027        Dst.percentLDDT12 = Src.percentLDDT12
1028        Dst.percentLDGT3 = Src.percentLDGT3
1029        Dst.percentLDGT4 = Src.percentLDGT4
1030        Dst.percentLDDT34 = Src.percentLDDT34
1031        Dst.gapTestOnRoad = Src.gapTestOnRoad
1032        Dst.fuelTankLT6000 = Src.fuelTankLT6000
1033        Dst.fuelTankLT60008500 = Src.fuelTankLT60008500
1034        Dst.fuelTankLT8500 = Src.fuelTankLT8500
1035        Dst.fuelTankCars = Src.fuelTankCars
1036        Dst.reboundEffect = Src.reboundEffect
1037        Dst.discountRate = Src.discountRate
1038        Dst.monopsonyCost = Src.monopsonyCost
1039        Dst.priceShockCost = Src.priceShockCost
1040        Dst.militarySecurityCost = Src.militarySecurityCost
1041        Dst.totalEconomicCost = Src.totalEconomicCost
1042        Dst.totalEconomicCostBBL = Src.totalEconomicCostBBL
1043        Dst.congestionCost = Src.congestionCost
1044        Dst.accidentCost = Src.accidentCost
1045        Dst.noiseCost = Src.noiseCost
1046        Dst.totalExternalCost = Src.totalExternalCost
1047        Dst.damageCostCO = Src.damageCostCO
1048        Dst.damageCostVOC = Src.damageCostVOC
1049        Dst.damageCostNOX = Src.damageCostNOX
1050        Dst.damageCostPM = Src.damageCostPM
1051        Dst.damageCostSO2 = Src.damageCostSO2
1052        Dst.damageCostC = Src.damageCostC
1053        Dst.damageCostCO2 = Src.damageCostCO2
1054        Dst.travelValuePerson = Src.travelValuePerson
1055        Dst.averageVehicleOccupancy = Src.averageVehicleOccupancy
1056        Dst.travelValueVehicle = Src.travelValueVehicle
1057
1058        i = 0
1059        For j = 0 To 25
1060            Dst.gasSalesAuto(i, j) = Src.gasSalesAuto(i, j)
1061            Dst.diesSalesAuto(i, j) = Src.diesSalesAuto(i, j)
1062            Dst.ethSalesAuto(i, j) = Src.ethSalesAuto(i, j)
1063            Dst.totSalesAuto(i, j) = Src.totSalesAuto(i, j)
1064            Dst.gasSalesLT(i, j) = Src.gasSalesLT(i, j)
1065            Dst.diesSalesLT(i, j) = Src.diesSalesLT(i, j)
1066            Dst.ethSalesLT(i, j) = Src.ethSalesLT(i, j)
1067            Dst.totSalesLT(i, j) = Src.totSalesLT(i, j)
1068            Dst.ratingEPAAuto(i, j) = Src.ratingEPAAuto(i, j)
1069            Dst.ratingEPALT(i, j) = Src.ratingEPALT(i, j)
```

```
1070            Dst.ratioMPGAuto(i, j) = Src.ratioMPGAuto(i, j)
1071            Dst.ratioMPGLT(i, j) = Src.ratioMPGLT(i, j)
1072        Next j
1073
1074        For i = 0 To 25
1075            Dst.fuelPriceGas(i) = Src.fuelPriceGas(i)
1076            Dst.fuelPriceDiesel(i) = Src.fuelPriceDiesel(i)
1077            Dst.fuelTaxGasFed(i) = Src.fuelTaxGasFed(i)
1078            Dst.fuelTaxGasState(i) = Src.fuelTaxGasState(i)
1079            Dst.fuelTaxGasTot(i) = Src.fuelTaxGasTot(i)
1080            Dst.fuelTaxDiesFed(i) = Src.fuelTaxDiesFed(i)
1081            Dst.fuelTaxDiesState(i) = Src.fuelTaxDiesState(i)
1082            Dst.fuelTaxDiesTot(i) = Src.fuelTaxDiesTot(i)
1083            Dst.fuelPretaxGas(i) = Src.fuelPretaxGas(i)
1084            Dst.fuelPretaxDies(i) = Src.fuelPretaxDies(i)
1085        Next i
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
```

```
1139        Dst.RatesMinCY = Src.RatesMinCY
1140        Dst.RatesMaxCY = Src.RatesMaxCY
1141
1142        Dim MaxMYIdx As Integer = CAFE.GMYI(Dst.RatesMaxMY)
1143        Dim MaxCYIdx As Integer = CAFE.GMYI(Dst.RatesMaxCY)
1144
1145        ' resize the variables
1146        ReDim Dst.coGasLDGV(MaxMYIdx, MaxCYIdx)
1147        ReDim Dst.coGasLDGT1(MaxMYIdx, MaxCYIdx)
1148        ReDim Dst.coGasLDGT2(MaxMYIdx, MaxCYIdx)
1149        ReDim Dst.coGasHDGV2b(MaxMYIdx, MaxCYIdx)
1150
1151        ReDim Dst.coGasRFGLDGV(MaxMYIdx, MaxCYIdx)
1152        ReDim Dst.coGasRFGLDGT1(MaxMYIdx, MaxCYIdx)
1153        ReDim Dst.coGasRFGLDGT2(MaxMYIdx, MaxCYIdx)
1154        ReDim Dst.coGasRFGHDGV2b(MaxMYIdx, MaxCYIdx)
1155
1156        ReDim Dst.coDiesLDDV(MaxMYIdx, MaxCYIdx)
1157        ReDim Dst.coDiesLDDT1(MaxMYIdx, MaxCYIdx)
1158        ReDim Dst.coDiesLDDT2(MaxMYIdx, MaxCYIdx)
1159        ReDim Dst.coDiesHDDV2b(MaxMYIdx, MaxCYIdx)
1160
1161        ReDim Dst.vocGasLDGV(MaxMYIdx, MaxCYIdx)
1162        ReDim Dst.vocGasLDGT1(MaxMYIdx, MaxCYIdx)
1163        ReDim Dst.vocGasLDGT2(MaxMYIdx, MaxCYIdx)
1164        ReDim Dst.vocGasHDGV2b(MaxMYIdx, MaxCYIdx)
1165
1166        ReDim Dst.vocGasRFGLDGV(MaxMYIdx, MaxCYIdx)
1167        ReDim Dst.vocGasRFGLDGT1(MaxMYIdx, MaxCYIdx)
1168        ReDim Dst.vocGasRFGLDGT2(MaxMYIdx, MaxCYIdx)
1169        ReDim Dst.vocGasRFGHDGV2b(MaxMYIdx, MaxCYIdx)
1170
1171        ReDim Dst.vocDiesLDDV(MaxMYIdx, MaxCYIdx)
1172        ReDim Dst.vocDiesLDDT1(MaxMYIdx, MaxCYIdx)
1173        ReDim Dst.vocDiesLDDT2(MaxMYIdx, MaxCYIdx)
1174        ReDim Dst.vocDiesHDDV2b(MaxMYIdx, MaxCYIdx)
1175
1176        ReDim Dst.noxGasLDGV(MaxMYIdx, MaxCYIdx)
1177        ReDim Dst.noxGasLDGT1(MaxMYIdx, MaxCYIdx)
1178        ReDim Dst.noxGasLDGT2(MaxMYIdx, MaxCYIdx)
1179        ReDim Dst.noxGasHDGV2b(MaxMYIdx, MaxCYIdx)
1180
1181        ReDim Dst.noxGasRFGLDGV(MaxMYIdx, MaxCYIdx)
1182        ReDim Dst.noxGasRFGLDGT1(MaxMYIdx, MaxCYIdx)
1183        ReDim Dst.noxGasRFGLDGT2(MaxMYIdx, MaxCYIdx)
1184        ReDim Dst.noxGasRFGHDGV2b(MaxMYIdx, MaxCYIdx)
1185
1186        ReDim Dst.noxDiesLDDV(MaxMYIdx, MaxCYIdx)
1187        ReDim Dst.noxDiesLDDT1(MaxMYIdx, MaxCYIdx)
1188        ReDim Dst.noxDiesLDDT2(MaxMYIdx, MaxCYIdx)
1189        ReDim Dst.noxDiesHDDV2b(MaxMYIdx, MaxCYIdx)
1190
1191        ReDim Dst.pmGasLDGV(MaxMYIdx, MaxCYIdx)
1192        ReDim Dst.pmGasLDGT1(MaxMYIdx, MaxCYIdx)
1193        ReDim Dst.pmGasLDGT2(MaxMYIdx, MaxCYIdx)
1194        ReDim Dst.pmGasHDGV2b(MaxMYIdx, MaxCYIdx)
1195
1196        ReDim Dst.pmGasRFGLDGV(MaxMYIdx, MaxCYIdx)
1197        ReDim Dst.pmGasRFGLDGT1(MaxMYIdx, MaxCYIdx)
1198        ReDim Dst.pmGasRFGLDGT2(MaxMYIdx, MaxCYIdx)
1199        ReDim Dst.pmGasRFGHDGV2b(MaxMYIdx, MaxCYIdx)
1200
1201        ReDim Dst.pmDiesLDDV(MaxMYIdx, MaxCYIdx)
1202        ReDim Dst.pmDiesLDDT1(MaxMYIdx, MaxCYIdx)
1203        ReDim Dst.pmDiesLDDT2(MaxMYIdx, MaxCYIdx)
1204        ReDim Dst.pmDiesHDDV2b(MaxMYIdx, MaxCYIdx)
1205
1206        ReDim Dst.so2GasLDGV(MaxMYIdx, MaxCYIdx)
1207        ReDim Dst.so2GasLDGT1(MaxMYIdx, MaxCYIdx)
```

```
1208        ReDim Dst.so2GasLDGT2(MaxMYIdx, MaxCYIdx)
1209        ReDim Dst.so2GasHDGV2b(MaxMYIdx, MaxCYIdx)
1210
1211        ReDim Dst.so2GasRFGLDGV(MaxMYIdx, MaxCYIdx)
1212        ReDim Dst.so2GasRFGLDGT1(MaxMYIdx, MaxCYIdx)
1213        ReDim Dst.so2GasRFGLDGT2(MaxMYIdx, MaxCYIdx)
1214        ReDim Dst.so2GasRFGHDGV2b(MaxMYIdx, MaxCYIdx)
1215
1216        ReDim Dst.so2DiesLDDV(MaxMYIdx, MaxCYIdx)
1217        ReDim Dst.so2DiesLDDT1(MaxMYIdx, MaxCYIdx)
1218        ReDim Dst.so2DiesLDDT2(MaxMYIdx, MaxCYIdx)
1219        ReDim Dst.so2DiesHDDV2b(MaxMYIdx, MaxCYIdx)
1220
1221        For i = 0 To MaxMYIdx
1222           For j = 0 To MaxCYIdx
1223              Dst.coGasLDGV(i, j) = Src.coGasLDGV(i, j)
1224              Dst.coGasLDGT1(i, j) = Src.coGasLDGT1(i, j)
1225              Dst.coGasLDGT2(i, j) = Src.coGasLDGT2(i, j)
1226              Dst.coGasHDGV2b(i, j) = Src.coGasHDGV2b(i, j)
1227
1228              Dst.coGasRFGLDGV(i, j) = Src.coGasRFGLDGV(i, j)
1229              Dst.coGasRFGLDGT1(i, j) = Src.coGasRFGLDGT1(i, j)
1230              Dst.coGasRFGLDGT2(i, j) = Src.coGasRFGLDGT2(i, j)
1231              Dst.coGasRFGHDGV2b(i, j) = Src.coGasRFGHDGV2b(i, j)
1232
1233              Dst.coDiesLDDV(i, j) = Src.coDiesLDDV(i, j)
1234              Dst.coDiesLDDT1(i, j) = Src.coDiesLDDT1(i, j)
1235              Dst.coDiesLDDT2(i, j) = Src.coDiesLDDT2(i, j)
1236              Dst.coDiesHDDV2b(i, j) = Src.coDiesHDDV2b(i, j)
1237
1238              Dst.vocGasLDGV(i, j) = Src.vocGasLDGV(i, j)
1239              Dst.vocGasLDGT1(i, j) = Src.vocGasLDGT1(i, j)
1240              Dst.vocGasLDGT2(i, j) = Src.vocGasLDGT2(i, j)
1241              Dst.vocGasHDGV2b(i, j) = Src.vocGasHDGV2b(i, j)
1242
1243              Dst.vocGasRFGLDGV(i, j) = Src.vocGasRFGLDGV(i, j)
1244              Dst.vocGasRFGLDGT1(i, j) = Src.vocGasRFGLDGT1(i, j)
1245              Dst.vocGasRFGLDGT2(i, j) = Src.vocGasRFGLDGT2(i, j)
1246              Dst.vocGasRFGHDGV2b(i, j) = Src.vocGasRFGHDGV2b(i, j)
1247
1248              Dst.vocDiesLDDV(i, j) = Src.vocDiesLDDV(i, j)
1249              Dst.vocDiesLDDT1(i, j) = Src.vocDiesLDDT1(i, j)
1250              Dst.vocDiesLDDT2(i, j) = Src.vocDiesLDDT2(i, j)
1251              Dst.vocDiesHDDV2b(i, j) = Src.vocDiesHDDV2b(i, j)
1252
1253              Dst.noxGasLDGV(i, j) = Src.noxGasLDGV(i, j)
1254              Dst.noxGasLDGT1(i, j) = Src.noxGasLDGT1(i, j)
1255              Dst.noxGasLDGT2(i, j) = Src.noxGasLDGT2(i, j)
1256              Dst.noxGasHDGV2b(i, j) = Src.noxGasHDGV2b(i, j)
1257
1258              Dst.noxGasRFGLDGV(i, j) = Src.noxGasRFGLDGV(i, j)
1259              Dst.noxGasRFGLDGT1(i, j) = Src.noxGasRFGLDGT1(i, j)
1260              Dst.noxGasRFGLDGT2(i, j) = Src.noxGasRFGLDGT2(i, j)
1261              Dst.noxGasRFGHDGV2b(i, j) = Src.noxGasRFGHDGV2b(i, j)
1262
1263              Dst.noxDiesLDDV(i, j) = Src.noxDiesLDDV(i, j)
1264              Dst.noxDiesLDDT1(i, j) = Src.noxDiesLDDT1(i, j)
1265              Dst.noxDiesLDDT2(i, j) = Src.noxDiesLDDT2(i, j)
1266              Dst.noxDiesHDDV2b(i, j) = Src.noxDiesHDDV2b(i, j)
1267
1268              Dst.pmGasLDGV(i, j) = Src.pmGasLDGV(i, j)
1269              Dst.pmGasLDGT1(i, j) = Src.pmGasLDGT1(i, j)
1270              Dst.pmGasLDGT2(i, j) = Src.pmGasLDGT2(i, j)
1271              Dst.pmGasHDGV2b(i, j) = Src.pmGasHDGV2b(i, j)
1272
1273              Dst.pmGasRFGLDGV(i, j) = Src.pmGasRFGLDGV(i, j)
1274              Dst.pmGasRFGLDGT1(i, j) = Src.pmGasRFGLDGT1(i, j)
1275              Dst.pmGasRFGLDGT2(i, j) = Src.pmGasRFGLDGT2(i, j)
1276              Dst.pmGasRFGHDGV2b(i, j) = Src.pmGasRFGHDGV2b(i, j)
```

```
1277
1278                Dst.pmDiesLDDV(i, j) = Src.pmDiesLDDV(i, j)
1279                Dst.pmDiesLDDT1(i, j) = Src.pmDiesLDDT1(i, j)
1280                Dst.pmDiesLDDT2(i, j) = Src.pmDiesLDDT2(i, j)
1281                Dst.pmDiesHDDV2b(i, j) = Src.pmDiesHDDV2b(i, j)
1282
1283                Dst.so2GasLDGV(i, j) = Src.so2GasLDGV(i, j)
1284                Dst.so2GasLDGT1(i, j) = Src.so2GasLDGT1(i, j)
1285                Dst.so2GasLDGT2(i, j) = Src.so2GasLDGT2(i, j)
1286                Dst.so2GasHDGV2b(i, j) = Src.so2GasHDGV2b(i, j)
1287
1288                Dst.so2GasRFGLDGV(i, j) = Src.so2GasRFGLDGV(i, j)
1289                Dst.so2GasRFGLDGT1(i, j) = Src.so2GasRFGLDGT1(i, j)
1290                Dst.so2GasRFGLDGT2(i, j) = Src.so2GasRFGLDGT2(i, j)
1291                Dst.so2GasRFGHDGV2b(i, j) = Src.so2GasRFGHDGV2b(i, j)
1292
1293                Dst.so2DiesLDDV(i, j) = Src.so2DiesLDDV(i, j)
1294                Dst.so2DiesLDDT1(i, j) = Src.so2DiesLDDT1(i, j)
1295                Dst.so2DiesLDDT2(i, j) = Src.so2DiesLDDT2(i, j)
1296                Dst.so2DiesHDDV2b(i, j) = Src.so2DiesHDDV2b(i, j)
1297            Next j
1298        Next i
1299    End Sub
1300
1301    '
1302    ' other methods
1303    '
1304    Public Function GetWillingnessToPayFines(ByVal MY As Integer, ByVal MfgIdx As Integer) ↙
        As Boolean
1305        Dim ThisMfg As String
1306        ThisMfg = CAFE.Mfgs(MfgIdx).MfgName
1307        Dim i As Integer
1308        Dim rCode As Boolean = False
1309        For i = 0 To Me.WillingToPayFinesMfgList.Length - 1
1310            If ThisMfg = Me.WillingToPayFinesMfgList(i) Then
1311                rCode = Me.WillingToPayFines(CAFE.GMYI(MY), i)
1312                Exit For
1313            End If
1314        Next
1315        Return rCode
1316    End Function
1317 End Class
1318
1319
1320 Public Class kMyTypeClass
1321
1322    '
1323    ' variables
1324    '
1325    Public Year As Integer
1326    Public Gasoline As Double
1327    Public Diesel As Double
1328
1329    Private Shared kMyParameters As ParametersClass
1330
1331
1332    '
1333    ' constructors
1334    '
1335    Public Sub New()
1336    End Sub
1337
1338    Public Sub New(ByVal Year As Integer, ByVal Gasoline As Double, ByVal Diesel As Double)
1339        Me.Year = Year
1340        Me.Gasoline = Gasoline
1341        Me.Diesel = Diesel
1342    End Sub
1343
1344    '
```

```
1345      ' copy methods
1346      '
1347      Public Sub Copy(ByRef Dst As kMyTypeClass)
1348          kMyTypeClass.Copy(Dst, Me)
1349      End Sub
1350
1351      Public Shared Sub Copy(ByRef Dst As kMyTypeClass, ByVal src As kMyTypeClass)
1352          Dst = New kMyTypeClass(src.Year, src.Gasoline, src.Diesel)
1353      End Sub
1354
1355      '
1356      ' other methods
1357      '
1358
1359      '''
1360      ' This function calculates and returns an array of kMyTypeClass objects for model years ↙
          2000
1361      ' up to MaxMY.
1362      '
1363      Public Shared Function GetKmyValues(ByVal Parameters As ParametersClass) As kMyTypeClass ↙
          ()
1364          kMyTypeClass.kMyParameters = Parameters
1365
1366          Dim i As Integer
1367          Dim iMaxYear As Integer = CAFE.GMYI(kMyParameters.wtpfMaxYear, False)
1368          Dim Kmy(iMaxYear) As kMyTypeClass
1369
1370          For i = 0 To iMaxYear
1371              Kmy(i) = GetKmyValue(CAFE.GetMYearFromIndex(i))
1372          Next i
1373          ' return the calculated Kmy() array
1374          Return Kmy
1375      End Function
1376
1377      '''
1378      ' This function calculates and returns a kMyTypeClass object for a given model year.
1379      '
1380      Public Shared Function GetKmyValue(ByVal MY As Integer) As kMyTypeClass
1381          ' initial checks
1382          If IsNothing(kMyParameters) Then
1383              ' can't calculate kMy values if there are no parameters
1384              Return Nothing
1385          End If
1386
1387          ' declare vars
1388          Dim SumGasoline As Double    ' stores the running sum of gas prices
1389          Dim SumDiesel As Double      ' stores the running sum of diesel prices
1390
1391          Dim v As Integer             ' vintage
1392          Dim p As Integer             ' payback
1393          Dim r As Double              ' discount rate
1394          Dim frac As Double           ' stores the fraction of a year
1395
1396          Dim iMY As Integer           ' index of the current MY + vintage
1397
1398          Dim Surv As Double           ' survivability rate after v years
1399          Dim Miles As Double          ' annual mileage driven after v years
1400
1401          Dim GasPrice As Double       ' the price of gasoline at a given MY
1402          Dim DieselPrice As Double    ' the price of diesel at a given MY
1403
1404          ' set initial vals
1405          SumGasoline = 0      ' set the sums initially to 0
1406          SumDiesel = 0        '
1407
1408          v = 0
1409          p = CInt(Math.Floor(kMyParameters.Payback))
1410          r = kMyParameters.discountRate
1411          ' set frac to be the payback period - floor(payback)
```

```
1412          ' e.g. the fraction of a year
1413          frac = kMyParameters.Payback - p
1414
1415          ' calculate Kmy
1416          For v = 0 To p
1417              ' set values for this iteration
1418              '
1419              ' the survivability and mileage values are obtained by taking the average of
1420              ' survivability and mileage values of individual mobile6 classes
1421              Surv = (kMyParameters.survivalLDGV(v, 0) + _
1422                      kMyParameters.survivalLDGT1(v, 0) + _
1423                      kMyParameters.survivalLDGT2(v, 0) + _
1424                      kMyParameters.survivalHDGV2b(v, 0) + _
1425                      kMyParameters.survivalLDDV(v, 0) + _
1426                      kMyParameters.survivalLDDV1(v, 0) + _
1427                      kMyParameters.survivalLDDV2(v, 0) + _
1428                      kMyParameters.survivalHDDV2b(v, 0)) / 8
1429              'Miles = (kMyParameters.milesLDGV(v, 0) + _
1430              '         kMyParameters.milesLDGT1(v, 0) + _
1431              '         kMyParameters.milesLDGT2(v, 0) + _
1432              '         kMyParameters.milesHDGV2b(v, 0) + _
1433              '         kMyParameters.milesLDDV(v, 0) + _
1434              '         kMyParameters.milesLDDV1(v, 0) + _
1435              '         kMyParameters.milesLDDV2(v, 0) + _
1436              '         kMyParameters.milesHDDV2b(v, 0)) / 8
1437              Miles = (kMyParameters.milesAuto(v, 0) + _
1438                       kMyParameters.milesPickup(v, 0) + _
1439                       kMyParameters.milesSuv(v, 0) + _
1440                       kMyParameters.milesVan(v, 0)) / 4
1441              ' TODO: if no auto miles values, divide by 3.
1442
1443              iMY = CAFE.GMYI(Math.Min(2025, MY + v), False)
1444
1445              GasPrice = kMyParameters.fuelPriceGas(iMY)
1446              DieselPrice = kMyParameters.fuelPriceDiesel(iMY)
1447
1448              SumGasoline += ((Surv * Miles * GasPrice / (1 + r) ^ v))
1449              SumDiesel += ((Surv * Miles * DieselPrice / (1 + r) ^ v))
1450          Next v  ' increase vintage
1451          ' take into account the fraction of a year
1452          SumGasoline += (frac * (Surv * Miles * GasPrice / (1 + r) ^ v))
1453          SumDiesel += (frac * (Surv * Miles * DieselPrice / (1 + r) ^ v))
1454
1455          ' finalize...create new Kmy instance and return it
1456          Return (New kMyTypeClass(MY, SumGasoline, SumDiesel))
1457      End Function
1458
1459  End Class
1460
1461
```

```
Public Class PbDataClass
    Private m_Reference As Double
    Private m_Low As Double
    Private m_High As Double
    Public Sub New()
    End Sub
    Public Property Reference() As Double
        Get
            Return m_Reference
        End Get
        Set(ByVal Value As Double)
            m_Reference = Value
        End Set
    End Property
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set(ByVal Value As Double)
            m_Low = Value
        End Set
    End Property
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set(ByVal Value As Double)
            m_High = Value
        End Set
    End Property
End Class

```

```
Public Class PbTypeClass
    Private m_MY As Integer
    Private m_Gasoline As PbDataClass
    Private m_Diesel As PbDataClass
    Public Sub New()
        m_Gasoline = New PbDataClass
        m_Diesel = New PbDataClass
    End Sub
    Public Property MY() As Integer
        Get
            Return m_MY
        End Get
        Set(ByVal Value As Integer)
            m_MY = Value
        End Set
    End Property
    Public Property Gasoline() As PbDataClass
        Get
            Return m_Gasoline
        End Get
        Set(ByVal Value As PbDataClass)
            m_Gasoline.High = Value.High
            m_Gasoline.Low = Value.Low
            m_Gasoline.Reference = Value.Reference
        End Set
    End Property
    Public Property diesel() As PbDataClass
        Get
            Return m_Diesel
        End Get
        Set(ByVal Value As PbDataClass)
            m_Diesel.High = Value.High
            m_Diesel.Low = Value.Low
            m_Diesel.Reference = Value.Reference
        End Set
    End Property
End Class

```

```
 1 Imports System
 2 Imports System.IO
 3 Imports System.Text
 4
 5 Public Class Profiling
 6
 7     Private fileName As String
 8     Private stream As FileStream
 9     Private ascii As ASCIIEncoding
10     Private myStack As Stack
11     Private buffer As String = ""
12
13     Private myProfile As New Profile
14     Private myProcess As Process
15     Private myProcessId As Integer
16
17     Public Shared DEBUG As Boolean = False
18     Public Shared GET_MEM_USAGE As Boolean = False
19
20     Public Sub New(Optional ByVal FileName As String = "profiler.csv")
21         If Profiling.DEBUG Then
22             If Not Directory.Exists(Path.GetDirectoryName(FileName)) Then
23                 Directory.CreateDirectory(Path.GetDirectoryName(FileName))
24             End If
25
26             Me.fileName = FileName
27             Me.stream = New FileStream(FileName, FileMode.Create, FileAccess.Write,
    FileShare.None)
28             Me.ascii = New ASCIIEncoding
29             Me.myStack = New Stack
30             Me.myProcessId = Process.GetCurrentProcess().Id
31             'Me.myProcess = Process.GetProcessesByName("CAFE_Reform").GetValue(0)
32
33             Dim str As String = ",Class Name,Sub/Function Name,Run Time (ms)"
34             If GET_MEM_USAGE Then
35                 str += ",Mem Before (KB),Mem After (KB),Mem Delta (KB),VMem Before (KB),
    VMem After (KB),VMem Delta (KB)"
36             End If
37             Dim bytes() As Byte = ascii.GetBytes(str + NewLine)
38             Me.stream.Write(bytes, 0, bytes.Length)
39         End If
40     End Sub
41
42     Public Sub FunctionStart()
43         If Profiling.DEBUG Then
44             Me.myProfile.startTime = Now.Ticks
45
46             If Profiling.GET_MEM_USAGE Then
47                 Me.myProcess = Process.GetProcessById(Me.myProcessId)
48
49                 Me.myProfile.memUsage = Me.myProcess.WorkingSet
50                 Me.myProfile.vMemUsage = Me.myProcess.PagedMemorySize
51                 'Me.myProfile.pMemUsage = Me.myProcess.PrivateMemorySize
52             End If
53
54             Me.myStack.Push(Me.myProfile)
55         End If
56     End Sub
57
58     Public Sub FunctionEnd(ByVal fnc As Reflection.MethodBase)
59         If Profiling.DEBUG Then
60             Dim str As String
61             Dim depth As Integer = Me.myStack.Count
62
63             Dim myProfile As Profile = Me.myStack.Pop()
64             Dim deltaTime As Long = (Now.Ticks - myProfile.startTime) / 10000
65
66             str = (depth & "," & fnc.ReflectedType.Name & "," & fnc.Name & "," & deltaTime)
67
68             If Profiling.GET_MEM_USAGE Then
```

```
69                  Me.myProcess = Process.GetProcessById(Me.myProcessId)
70
71                  Dim initialMem As Long = myProfile.memUsage / 1024
72                  Dim finalMem As Long = Me.myProcess.WorkingSet / 1024
73                  Dim deltaMem As Long = finalMem - initialMem
74
75                  Dim initialVMem As Long = myProfile.vMemUsage / 1024
76                  Dim finalVMem As Long = Me.myProcess.PagedMemorySize / 1024
77                  Dim deltaVMem As Long = finalVMem - initialVMem
78
79                  'Dim initialPMem As Long = myProfile.pMemUsage / 1024
80                  'Dim finalPMem As Long = Me.myProcess.PrivateMemorySize / 1024
81                  'Dim deltaPMem As Long = finalPMem - initialPMem
82
83                  str += ("," & initialMem & "," & finalMem & "," & deltaMem & _
84                          "," & initialVMem & "," & finalVMem & "," & deltaVMem)
85              End If
86
87              str += NewLine
88
89              'Me.buffer += str
90
91              Dim bytes() As Byte = ascii.GetBytes(str)
92              Me.stream.Write(bytes, 0, bytes.Length)
93              'Me.stream.WriteByte(AscW(Chr(13)))
94              'Me.stream.WriteByte(AscW(Chr(10)))
95              'Me.stream.Write(ascii.GetBytes(vbCrLf), 0, 2)
96          End If
97      End Sub
98
99      Public Sub OpenLast()
100         If Profiling.DEBUG Then
101             Try
102                 Me.stream = New FileStream(fileName, FileMode.OpenOrCreate, FileAccess.
        Write, FileShare.None)
103                 Me.stream.Seek(0, SeekOrigin.End)
104             Catch
105                 ' do nothing
106             End Try
107         End If
108     End Sub
109
110     Public Sub Close()
111         If Profiling.DEBUG Then
112             If Not IsNothing(Me.stream) Then
113                 Me.stream.Close()
114             End If
115
116             'Try
117             '    'Dim bytes() As Byte = Me.ascii.GetBytes(Me.buffer)
118             '    'Me.stream = New FileStream(Me.fileName, FileMode.Create, FileAccess.Write
        , FileShare.None)
119             '    'Me.stream.Write(bytes, 0, bytes.Length)
120             '    Me.stream.Close()
121             'Catch ex As Exception
122             '    MsgBox("Could not open file " & Me.fileName & ".  App profiling not
        written.")
123             'End Try
124         End If
125     End Sub
126
127 End Class
128
129 Public Structure Profile
130
131     Public startTime As Long
132     Public memUsage As Long         ' mem usage
133     Public vMemUsage As Long        ' virtual mem
134     Public pMemUsage As Long        ' private mem
135
```

```
136 End Structure
137
```

```
Imports System.Threading

Public Class ProgressIndicator
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        New_()
    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents m_ProgressBarOverAll As System.Windows.Forms.ProgressBar
    Friend WithEvents m_ProgressBarCurrent As System.Windows.Forms.ProgressBar
    Friend WithEvents m_OverAllLabel As System.Windows.Forms.Label
    Friend WithEvents m_CurrentLabel As System.Windows.Forms.Label
    Friend WithEvents m_DetailsButton As System.Windows.Forms.Button
    Friend WithEvents m_DetailsTextBox As System.Windows.Forms.TextBox
    Friend WithEvents m_RemainingTimeLabel As System.Windows.Forms.Label
    Friend WithEvents m_RemaingTimeCounterLabel As System.Windows.Forms.Label
    Friend WithEvents m_ATRTimer As System.Windows.Forms.Timer
    Friend WithEvents m_CloseButton As System.Windows.Forms.Button
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Me.components = New System.ComponentModel.Container
        Me.m_ProgressBarOverAll = New System.Windows.Forms.ProgressBar
        Me.m_ProgressBarCurrent = New System.Windows.Forms.ProgressBar
        Me.m_OverAllLabel = New System.Windows.Forms.Label
        Me.m_CurrentLabel = New System.Windows.Forms.Label
        Me.m_DetailsButton = New System.Windows.Forms.Button
        Me.m_DetailsTextBox = New System.Windows.Forms.TextBox
        Me.m_RemainingTimeLabel = New System.Windows.Forms.Label
        Me.m_RemaingTimeCounterLabel = New System.Windows.Forms.Label
        Me.m_ATRTimer = New System.Windows.Forms.Timer(Me.components)
        Me.m_CloseButton = New System.Windows.Forms.Button
        Me.SuspendLayout()
        '
        'm_ProgressBarOverAll
        '
        Me.m_ProgressBarOverAll.Location = New System.Drawing.Point(104, 16)
        Me.m_ProgressBarOverAll.Name = "m_ProgressBarOverAll"
        Me.m_ProgressBarOverAll.Size = New System.Drawing.Size(400, 23)
        Me.m_ProgressBarOverAll.Step = 1
        Me.m_ProgressBarOverAll.TabIndex = 1
        '
        'm_ProgressBarCurrent
        '
        Me.m_ProgressBarCurrent.Location = New System.Drawing.Point(104, 48)
        Me.m_ProgressBarCurrent.Name = "m_ProgressBarCurrent"
        Me.m_ProgressBarCurrent.Size = New System.Drawing.Size(400, 23)
```

```
71          Me.m_ProgressBarCurrent.Step = 1
72          Me.m_ProgressBarCurrent.TabIndex = 3
73          '
74          'm_OverAllLabel
75          '
76          Me.m_OverAllLabel.Location = New System.Drawing.Point(16, 16)
77          Me.m_OverAllLabel.Name = "m_OverAllLabel"
78          Me.m_OverAllLabel.Size = New System.Drawing.Size(96, 23)
79          Me.m_OverAllLabel.TabIndex = 0
80          Me.m_OverAllLabel.Text = "Overall progress:"
81          Me.m_OverAllLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
82          '
83          'm_CurrentLabel
84          '
85          Me.m_CurrentLabel.Location = New System.Drawing.Point(16, 48)
86          Me.m_CurrentLabel.Name = "m_CurrentLabel"
87          Me.m_CurrentLabel.Size = New System.Drawing.Size(96, 23)
88          Me.m_CurrentLabel.TabIndex = 2
89          Me.m_CurrentLabel.Text = "Current action:"
90          Me.m_CurrentLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
91          '
92          'm_DetailsButton
93          '
94          Me.m_DetailsButton.Location = New System.Drawing.Point(405, 75)
95          Me.m_DetailsButton.Name = "m_DetailsButton"
96          Me.m_DetailsButton.Size = New System.Drawing.Size(100, 20)
97          Me.m_DetailsButton.TabIndex = 5
98          Me.m_DetailsButton.Text = "Show Details >>"
99          '
100         'm_DetailsTextBox
101         '
102         Me.m_DetailsTextBox.AcceptsTab = True
103         Me.m_DetailsTextBox.BackColor = System.Drawing.SystemColors.Window
104         Me.m_DetailsTextBox.Location = New System.Drawing.Point(16, 104)
105         Me.m_DetailsTextBox.Multiline = True
106         Me.m_DetailsTextBox.Name = "m_DetailsTextBox"
107         Me.m_DetailsTextBox.ReadOnly = True
108         Me.m_DetailsTextBox.ScrollBars = System.Windows.Forms.ScrollBars.Both
109         Me.m_DetailsTextBox.Size = New System.Drawing.Size(488, 208)
110         Me.m_DetailsTextBox.TabIndex = 6
111         Me.m_DetailsTextBox.Text = ""
112         Me.m_DetailsTextBox.Visible = False
113         Me.m_DetailsTextBox.WordWrap = False
114         '
115         'm_RemainingTimeLabel
116         '
117         Me.m_RemainingTimeLabel.Location = New System.Drawing.Point(16, 78)
118         Me.m_RemainingTimeLabel.Name = "m_RemainingTimeLabel"
119         Me.m_RemainingTimeLabel.Size = New System.Drawing.Size(160, 16)
120         Me.m_RemainingTimeLabel.TabIndex = 7
121         Me.m_RemainingTimeLabel.Text = "Approximate Time Remaining:"
122         '
123         'm_RemaingTimeCounterLabel
124         '
125         Me.m_RemaingTimeCounterLabel.Location = New System.Drawing.Point(192, 78)
126         Me.m_RemaingTimeCounterLabel.Name = "m_RemaingTimeCounterLabel"
127         Me.m_RemaingTimeCounterLabel.Size = New System.Drawing.Size(80, 16)
128         Me.m_RemaingTimeCounterLabel.TabIndex = 8
129         Me.m_RemaingTimeCounterLabel.Text = "Estimating..."
130         Me.m_RemaingTimeCounterLabel.TextAlign = System.Drawing.ContentAlignment.TopRight
131         '
132         'm_ATRTimer
133         '
134         Me.m_ATRTimer.Interval = 1000
135         '
136         'm_CloseButton
137         '
138         Me.m_CloseButton.Location = New System.Drawing.Point(296, 75)
139         Me.m_CloseButton.Name = "m_CloseButton"
140         Me.m_CloseButton.Size = New System.Drawing.Size(100, 20)
```

```
141         Me.m_CloseButton.TabIndex = 4
142         Me.m_CloseButton.Text = "&Close"
143         '
144         'ProgressIndicator
145         '
146         Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
147         Me.ClientSize = New System.Drawing.Size(522, 346)
148         Me.ControlBox = False
149         Me.Controls.Add(Me.m_CloseButton)
150         Me.Controls.Add(Me.m_RemaingTimeCounterLabel)
151         Me.Controls.Add(Me.m_RemainingTimeLabel)
152         Me.Controls.Add(Me.m_DetailsTextBox)
153         Me.Controls.Add(Me.m_DetailsButton)
154         Me.Controls.Add(Me.m_ProgressBarCurrent)
155         Me.Controls.Add(Me.m_ProgressBarOverAll)
156         Me.Controls.Add(Me.m_CurrentLabel)
157         Me.Controls.Add(Me.m_OverAllLabel)
158         Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow
159         Me.MaximizeBox = False
160         Me.MinimizeBox = False
161         Me.Name = "ProgressIndicator"
162         Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
163         Me.Text = "Please Wait..."
164         Me.ResumeLayout(False)
165
166     End Sub
167
168 #End Region
169
170     Private startTime, endTime As Long
171     Private elapsedTime As Long
172
173     Private buffer As String = ""
174     Private FirstLine As String = ""
175     Private DetailsVisible As Boolean = False
176
177     Private OnePercentMax As Double
178     Private OnePercentEstimate As Double
179     Private OnePercentStartTime As Long
180     Private OnePercentEndTime As Long
181     Private OnePercentEstimated As Boolean = False
182
183     Private TenPercentMax As Double
184     Private TenPercentEstimate As Double
185     Private TenPercentStartTime As Long
186     Private TenPercentEndTime As Long
187     Private TenPercentEstimated As Boolean = False
188     Private TenPercentEstimateMultiplier As Double
189
190     Private ApproximateTimeRemaining As Long
191
192     Private Const NewLine As String = Chr(13) + Chr(10)
193
194     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
195     ' methods for user interaction
196
197     Friend Sub init(Optional ByVal title As String = Nothing)
198         Me.setup(title)
199         Me.display()
200     End Sub
201
202     Friend Sub terminate()
203         Debug.WriteLine(Now)
204         Me.endTime = Now.Ticks
205         Me.elapsedTime = (Me.endTime - Me.startTime) / 10000
206         Debug.WriteLine("Elapsed time: " & elapsedTime & " ms.")
207
208         Me.m_RemaingTimeCounterLabel.Text = "0 s."
209         Me.m_ATRTimer.Stop()
210         Me.m_ATRTimer.Enabled = False
```

```
211
212         ' hide and close the window
213         Me.Visible = False
214         Me.Hide()
215         Me.Close()
216     End Sub
217
218     Friend Sub PerformStep(Optional ByVal stepDetails As String = Nothing)
219         Me.m_ProgressBarOverAll.PerformStep()
220         Me.m_ProgressBarCurrent.PerformStep()
221         Me.WriteDetails(stepDetails)
222         Application.DoEvents()
223         'Me.Refresh()            ' call to refresh probably not necessary
224     End Sub
225
226     Friend Sub WriteDetails(ByVal stepDetails As String)
227         ' stepDetails cannot be nothing, empty, or blank
228         If Not IsNothing(stepDetails) OrElse _
229                stepDetails <> String.Empty OrElse _
230                stepDetails <> "" Then
231             '
232             Me.buffer = stepDetails & NewLine & buffer
233
234             If Me.buffer.Length > 10000 Then
235                 Me.buffer = Me.buffer.Substring(0, Me.buffer.LastIndexOf(NewLine))
236             End If
237
238             Me.m_DetailsTextBox.Text = buffer
239         End If
240
241         EstimateTimeRemaining()
242     End Sub
243
244     Private Sub EstimateTimeRemaining()
245         EstimateTimeRemaining(Me.OnePercentMax, Me.OnePercentEstimate, _
246                               Me.OnePercentStartTime, Me.OnePercentEndTime, _
247                               100, Me.OnePercentEstimated)
248
249         EstimateTimeRemaining(Me.TenPercentMax, Me.TenPercentEstimate, _
250                               Me.TenPercentStartTime, Me.TenPercentEndTime, _
251                               Me.TenPercentEstimateMultiplier, Me.TenPercentEstimated)
252     End Sub
253
254     Private Sub EstimateTimeRemaining(ByRef MaxVal As Double, _
255                                       ByRef Estimate As Double, _
256                                       ByRef StartTime As Long, _
257                                       ByRef EndTime As Long, _
258                                       ByRef EstimateMultiplier As Double, _
259                                       ByRef Estimated As Boolean)
260         '
261         If Not Estimated Then
262             If (Me.m_ProgressBarOverAll.Value) >= MaxVal Then
263                 ' disable and stop the timer (in case it was running)
264                 Me.m_ATRTimer.Enabled = False
265                 Me.m_ATRTimer.Stop()
266
267                 EndTime = Now.Ticks
268                 Estimate = (EndTime - StartTime) / 10000000
269                 ' adjust the estimate for the remaining percent
270                 Estimate *= EstimateMultiplier
271                 Estimated = True
272
273                 Me.ApproximateTimeRemaining = Estimate
274
275                 ' enable and start the timer
276                 Me.m_ATRTimer.Enabled = True
277                 Me.m_ATRTimer.Start()
278
279                 Debug.WriteLine((100 / EstimateMultiplier) & "% estimate produced: " &
    Estimate)
```

```
280             End If
281         Else
282             If EstimateMultiplier <= 10 Then
283                 ' 10% estimate interval
284                 ' from now on we'll be updating the estimate every 10%
285                 StartTime = Now.Ticks
286                 EstimateMultiplier -= 1
287                 MaxVal = MaxVal + 0.1 * Me.m_ProgressBarOverAll.Maximum
288                 Estimated = False
289             End If
290         End If
291     End Sub
292
293     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
294     ' methods dealing with setup, display, and initialization
295
296     ' extension to the constructor
297     Private Sub New_()
298         HideDetails()
299     End Sub
300
301     Private Sub setup(ByVal title As String)
302         Debug.WriteLine(Now)
303         Me.startTime = Now.Ticks
304         Me.m_ProgressBarOverAll.Value = 0
305         Me.m_ProgressBarCurrent.Value = 0
306         Me.OnePercentMax = 0.01 * Me.m_ProgressBarOverAll.Maximum
307         Me.TenPercentMax = 0.1 * Me.m_ProgressBarOverAll.Maximum
308         Me.TenPercentEstimateMultiplier = 10
309         Me.OnePercentStartTime = Now.Ticks
310         Me.TenPercentStartTime = Now.Ticks
311
312         ' clear the details textbox
313         Me.m_DetailsTextBox.Text = ""
314
315         If Not IsNothing(title) OrElse title <> "" OrElse title <> String.Empty Then
316             Me.Text = title.Trim()
317         End If
318     End Sub
319
320     Private Sub display()
321         'Me.ShowDialog()
322         Me.Show()
323     End Sub
324
325     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
326     ' methods dealing with the details displaying
327
328     Private Sub OnDetailsButton(ByVal sender As System.Object, ByVal e As System.EventArgs) ↙
         Handles m_DetailsButton.Click
329         If Me.DetailsVisible Then       ' details are currently visible
330             HideDetails()
331         Else                            ' details are not visible
332             ShowDetails()
333         End If
334     End Sub
335
336     Private Sub ShowDetails()
337         Me.Height = 368
338         Me.m_DetailsButton.Text = "Hide Details <<"
339         Me.m_DetailsTextBox.Visible = True
340         Me.DetailsVisible = True
341         Me.m_RemainingTimeLabel.Top = 320
342         Me.m_RemaingTimeCounterLabel.Top = 320
343         Me.Invalidate()
344         Me.Refresh()
345     End Sub
346
347     Private Sub HideDetails()
348         Me.Height = 120
```

```
349         Me.m_DetailsButton.Text = "Show Details >>"
350         Me.m_DetailsTextBox.Visible = False
351         Me.DetailsVisible = False
352         Me.m_RemainingTimeLabel.Top = 80
353         Me.m_RemaingTimeCounterLabel.Top = 80
354         Me.Invalidate()
355         Me.Refresh()
356     End Sub
357
358     Private Sub OnATRTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) ↙
         Handles m_ATRTimer.Tick
359         If ApproximateTimeRemaining > 0 Then
360             ApproximateTimeRemaining -= 1
361         End If
362
363         Dim hour As Integer = 0
364         Dim min As Integer = 0
365         Dim sec As Integer = ApproximateTimeRemaining
366         Dim TimeString As String = ""
367
368         If sec >= 3600 Then
369             hour = sec \ 3600
370             sec -= (3600 * hour)
371             TimeString = hour & " h. "
372         End If
373
374         If sec > 60 Then
375             min = sec \ 60
376             sec -= (60 * min)
377             TimeString += (min & " m. ")
378         End If
379
380         TimeString += (sec & " s.")
381
382         m_RemaingTimeCounterLabel.Text = TimeString
383     End Sub
384
385     Private Sub OnCloseButton_Click(ByVal sender As System.Object, ByVal e As System. ↙
        EventArgs) Handles m_CloseButton.Click
386         ' hide and close the window
387         Me.Visible = False
388         Me.Hide()
389         Me.Close()
390     End Sub
391
392     Private Sub m_DetailsTextBox_TextChanged(ByVal sender As System.Object, ByVal e As ↙
        System.EventArgs) Handles m_DetailsTextBox.TextChanged
393
394     End Sub
395 End Class
396
```

```
Public Class RawEnginesClass
    Inherits RawInput

#Region "EngineColumns"
    Public Engine_CodeColNum As Integer
    Public ManufacturerColNum As Integer
    Public NameColNum As Integer
    Public OriginColNum As Integer
    Public FuelColNum As Integer
    Public EngineOilViscosityColNum As Integer
    Public CycleColNum As Integer
    Public AirFuelRatioColNum As Integer
    Public FuelSystemColNum As Integer
    Public AspirationColNum As Integer
    Public ValvetrainDesignColNum As Integer
    Public ValveActuationTimingColNum As Integer
    Public ValveLiftColNum As Integer
    Public CylindersColNum As Integer
    Public ConfigurationColNum As Integer
    Public ValvesCylinderColNum As Integer
    Public DeactivationColNum As Integer
    Public DisplacementColNum As Integer
    Public MinCompressionRatioColNum As Integer
    Public MaxCompressionRatioColNum As Integer
    Public HorsepowerColNum As Integer
    Public TorqueColNum As Integer
    Public EngineNotesColNum As Integer

    Public LUBColNum As Integer
    Public EFRColNum As Integer
    Public OHCColNum As Integer
    Public VVTColNum As Integer
    Public DISPColNum As Integer
    Public VVLTColNum As Integer
    Public SIDIColNum As Integer
    Public SUPColNum As Integer
    Public CVAColNum As Integer
    Public IVTColNum As Integer
    Public VCRColNum As Integer

    Public DSLcolnum As Integer
#End Region

    Public Sub New()
    End Sub

    Public Sub New(ByVal fName As String)
        ' Read engines file and store raw data
        Dim EnginesFile As New C_Cafe_InFile
        Dim RowCount As Integer
        Dim ColCount As Integer

        If Not EnginesFile.OpenFile(fName, 2) Then
            ' could not open file for any reason, prompt user and exit...
            MsgBox("An error has occured while opening the engines input file.  " & _
                   "Please try again.", MsgBoxStyle.Exclamation, "Error Opening File")
            Exit Sub
        End If

        RowCount = EnginesFile.GetNumRows()
        ColCount = EnginesFile.GetNumColumns(1)
        Wb = EnginesFile.GetWorkbookObject

        ' parse wb title
        If Not Me.ParseWbTitle("CAFE Engines Dataset file", 1.0) Then
            ' if wb title or version did not match desired ones
            ' close file and exit sub
            EnginesFile.CloseFile()
            EnginesFile = Nothing
            Exit Sub
```

```
71          End If
72
73          ' load the data
74          Me.LoadData(Wb.ActiveSheet, RowCount, ColCount)
75
76          ' Now close the file
77          EnginesFile.CloseFile()
78          EnginesFile = Nothing
79      End Sub
80
81      '''
82      ' Loads the data from the given worksheet.
83      '
84      Public Sub LoadData(ByVal ws As Excel.Worksheet, ByVal RowCount As Integer, ByVal
        ColCount As Integer)
85          Dim ActSheet As Excel.Worksheet
86          Dim rng0 As Excel.Range
87          Dim rng1 As Excel.Range
88          Dim i As Integer
89          Dim j As Integer
90          Dim r As Integer
91          Dim MsgString As String
92
93          ' Access the worksheet
94          ActSheet = ws
95
96          ' Headers first
97          Me.RawHeaders = ActSheet.Range("1:1").Resize.Value
98
99          ' Now the data
100         rng0 = ActSheet.Cells(2, 1)
101         rng1 = ActSheet.Cells(RowCount, ColCount)
102         Me.RawData = ActSheet.Range(rng0, rng1).Resize.Value
103
104         ' Parse the column headers
105         ReDim ColMatch(ColCount)
106         For i = 1 To ColCount
107             CheckColumnName("Engine Code", 1, i, Engine_CodeColNum)
108             CheckColumnName("Manufacturer", 1, i, ManufacturerColNum)
109             CheckColumnName("Name", 1, i, NameColNum)
110             CheckColumnName("Origin", 1, i, OriginColNum)
111             CheckColumnName("Fuel", 1, i, FuelColNum)
112             CheckColumnName("Engine Oil Viscosity", 1, i, EngineOilViscosityColNum)
113             CheckColumnName("Cycle", 1, i, CycleColNum)
114             CheckColumnName("Air/Fuel Ratio", 1, i, AirFuelRatioColNum)
115             CheckColumnName("Fuel System", 1, i, FuelSystemColNum)
116             CheckColumnName("Aspiration", 1, i, AspirationColNum)
117             CheckColumnName("Valvetrain Design", 1, i, ValvetrainDesignColNum)
118             CheckColumnName("Valve Actuation/Timing", 1, i, ValveActuationTimingColNum)
119             CheckColumnName("Valve Lift", 1, i, ValveLiftColNum)
120             CheckColumnName("Cylinders", 1, i, CylindersColNum)
121             CheckColumnName("Configuration", 1, i, ConfigurationColNum)
122             CheckColumnName("Valves/Cylinder", 1, i, ValvesCylinderColNum)
123             CheckColumnName("Deactivation", 1, i, DeactivationColNum)
124             CheckColumnName("Deactivation ", 1, i, DeactivationColNum)    ' Early versions
        had a space
125             CheckColumnName("Displacement", 1, i, DisplacementColNum)
126             CheckColumnName("Compression Ratio (Min)", 1, i, MinCompressionRatioColNum)
127             CheckColumnName("Compression Ratio (Max)", 1, i, MaxCompressionRatioColNum)
128             CheckColumnName("Horsepower", 1, i, HorsepowerColNum)
129             CheckColumnName("Torque", 1, i, TorqueColNum)
130             CheckColumnName("Engine Notes", 1, i, EngineNotesColNum)
131
132             CheckColumnName("LUB", 1, i, LUBColNum)
133             CheckColumnName("EFR", 1, i, EFRColNum)
134             CheckColumnName("OHC", 1, i, OHCColNum)
135             CheckColumnName("VVT", 1, i, VVTColNum)
136             CheckColumnName("DISP", 1, i, DISPColNum)
137             CheckColumnName("VVLT", 1, i, VVLTColNum)
138             CheckColumnName("SIDI", 1, i, SIDIColNum)
```

```
139             CheckColumnName("SUP", 1, i, SUPColNum)
140             CheckColumnName("CVA", 1, i, CVAColNum)
141             CheckColumnName("IVT", 1, i, IVTColNum)
142             CheckColumnName("VCR", 1, i, VCRColNum)
143             CheckColumnName("DSL", 1, i, DSLcolnum)
144         Next
145
146         ' Check if the required columns are missing
147         MsgString = ""
148         Me.CheckRequiredColumn(Me.Engine_CodeColNum, "     - Engine Code", MsgString)
149         Me.CheckRequiredColumn(Me.ManufacturerColNum, "    - Manufacturer", MsgString)
150         Me.CheckRequiredColumn(Me.NameColNum, "     - Name", MsgString)
151         Me.CheckRequiredColumn(Me.FuelColNum, "     - Fuel", MsgString)
152         Me.CheckRequiredColumn(Me.EngineOilViscosityColNum, "     - Engine Oil Viscosity",
    MsgString)
153         Me.CheckRequiredColumn(Me.CycleColNum, "    - Cycle", MsgString)
154         Me.CheckRequiredColumn(Me.ValvetrainDesignColNum, "     - Valvetrain Design",
    MsgString)
155         Me.CheckRequiredColumn(Me.ValveLiftColNum, "     - Valve Lift", MsgString)
156         Me.CheckRequiredColumn(Me.CylindersColNum, "     - Cylinders", MsgString)
157         Me.CheckRequiredColumn(Me.ValvesCylinderColNum, "     - Valves/Cylinder", MsgString)
158         Me.CheckRequiredColumn(Me.DeactivationColNum, "     - Deactivation", MsgString)
159
160         '' Now check for any missing column names
161         'MsgString = ""
162         'For i = 1 To ColCount
163         '    If ColMatch(i) = False Then
164         '        MsgString = MsgString + i.ToString + " "
165         '    End If
166         'Next i
167
168         If MsgString <> "" Then
169             MsgBox("Could not find match for the following column(s): " & NewLine & _
170                    NewLine & MsgString & NewLine & "Please select a different workbook.", _
171                    MsgBoxStyle.Exclamation, "Missing Columns")
172             Exit Sub
173         End If
174
175         ' Now step through the engines and record the essentials...
176         ReDim CAFE.MasterEngines(RowCount - 2)
177         For r = 2 To RowCount
178             i = r - 2
179             j = r - 1
180
181             CAFE.MasterEngines(i) = New EnginesClass
182
183             ' Validate entries (the ones which matter)
184             Dim Reasons As String = String.Empty
185             Dim Valid As Boolean = True
186
187             Dim EngineCode As Integer = GetInteger(j, Engine_CodeColNum)
188             Dim Manufacturer As String = GetString(j, ManufacturerColNum)
189             Dim Origin As String = GetString(j, OriginColNum)
190             Dim Fuel As String = GetString(j, FuelColNum)
191             Dim EngineOilViscosity As String = GetString(j, EngineOilViscosityColNum)
192             Dim Cycle As String = GetString(j, CycleColNum)
193             Dim Aspiration As String = GetString(j, AspirationColNum)
194             Dim ValvetrainDesign As String = GetString(j, ValvetrainDesignColNum)
195             Dim ValveActuatingTiming As String = GetString(j, ValveActuationTimingColNum)
196             Dim ValveLift As String = GetString(j, ValveLiftColNum)
197             Dim Cylinders As Double = GetDouble(j, CylindersColNum)
198             Dim ValvesCylinder As Integer = GetInteger(j, ValvesCylinderColNum)
199             Dim Deactivation As Double = GetDouble(j, DeactivationColNum)
200             Dim Displacement As Double = GetDouble(j, DisplacementColNum)
201
202             Valid = ValidateField(EngineCode, 0, Integer.MaxValue, "EngineCode", Reasons)
203             Valid = Valid And ValidateField(Manufacturer, "Manufacturer", Reasons)
204             Valid = Valid And ValidateField(Fuel, "Fuel", Reasons)
205             Valid = Valid And ValidateField(Cycle, "Cycle", Reasons)
206             Valid = Valid And ValidateField(Aspiration, "Asperation", Reasons)
```

```
207             Valid = Valid And ValidateField(Cylinders, 1, Double.PositiveInfinity, "Cylinders", Reasons)
208             Valid = Valid And ValidateField(ValvesCylinder, 1, Integer.MaxValue, "ValvesCylinder", Reasons)
209
210             'Valid = Valid And ValidateField(Origin, "Origin", Reasons)
211             'Valid = Valid And ValidateField(EngineOilViscosity, "EngineOilViscosity", Reasons)
212             'Valid = Valid And ValidateField(ValvetrainDesign, "Valvetraindesign", Reasons)
213             'Valid = Valid And ValidateField(ValveActuatingTiming, "ValveActuatingTiming", Reasons)
214             'Valid = Valid And ValidateField(ValveLift, "ValveLift", Reasons)
215             'Valid = Valid And ValidateField(Deactivation, 0, Double.PositiveInfinity, "Deactivation", Reasons)
216             'Valid = Valid And ValidateField(Displacement, 0, Double.PositiveInfinity, "Displacement", Reasons)
217
218             CAFE.MasterEngines(i).MyIndexNumber = i
219             CAFE.MasterEngines(i).Valid = Valid
220             CAFE.MasterEngines(i).ReasonForBeingInvalid = Reasons
221             CAFE.MasterEngines(i).Engine_Code = EngineCode
222             CAFE.MasterEngines(i).Manufacturer = Manufacturer
223             CAFE.MasterEngines(i).Name = GetString(j, NameColNum)
224             CAFE.MasterEngines(i).Origin = Origin
225             CAFE.MasterEngines(i).Fuel = Fuel
226             CAFE.MasterEngines(i).EngineOilViscosity = EngineOilViscosity
227             CAFE.MasterEngines(i).Cycle = Cycle
228             CAFE.MasterEngines(i).AirFuelRatio = GetDouble(j, AirFuelRatioColNum)
229             CAFE.MasterEngines(i).FuelSystem = GetString(j, FuelSystemColNum)
230             CAFE.MasterEngines(i).Aspiration = Aspiration
231             CAFE.MasterEngines(i).ValvetrainDesign = ValvetrainDesign
232             CAFE.MasterEngines(i).ValveActuationTiming = ValveActuatingTiming
233             CAFE.MasterEngines(i).ValveLift = ValveLift
234             CAFE.MasterEngines(i).Configuration = GetString(j, ConfigurationColNum)
235             CAFE.MasterEngines(i).Cylinders = Cylinders
236             CAFE.MasterEngines(i).ValvesCylinder = ValvesCylinder
237             CAFE.MasterEngines(i).Deactivation = Deactivation
238             CAFE.MasterEngines(i).Displacement = Displacement
239             ' temporary string
240             Dim str As String
241             ' min and max compression ratios need special handling
242             Dim CompressionRatio() As String = Nothing
243             ' first do min compression ratio
244             str = GetString(j, MinCompressionRatioColNum)
245             If Not IsNothing(str) Then
246                 CompressionRatio = str.Split(":")
247             End If
248             CAFE.MasterEngines(i).MinCompressionRatio = GetDouble(CompressionRatio, 0)
249             ' then do max compression ratio
250             str = GetString(j, MaxCompressionRatioColNum)
251             If Not IsNothing(str) Then
252                 CompressionRatio = str.Split(":")
253             End If
254             CAFE.MasterEngines(i).MaxCompressionRatio = GetDouble(CompressionRatio, 0)
255             '
256             ' horsepower col needs special handling
257             Dim Horsepower() As String = Nothing
258             str = GetString(j, HorsepowerColNum)
259             If Not IsNothing(str) Then
260                 Horsepower = str.Split("@")
261             End If
262             CAFE.MasterEngines(i).Horsepower_HP = GetDouble(Horsepower, 0)
263             If Horsepower.Length > 1 Then
264                 CAFE.MasterEngines(i).Horsepower_RPM = GetDouble(Horsepower, 1)
265             End If
266             ' torque col also needs special handling
267             Dim Torque() As String = Nothing
268             str = GetString(j, TorqueColNum)
269             If Not IsNothing(str) Then
270                 Torque = str.Split("@")
```

```
271             End If
272             CAFE.MasterEngines(i).Torque_HP = GetDouble(Torque, 0)
273             If Torque.Length > 1 Then
274                 CAFE.MasterEngines(i).Torque_RPM = GetDouble(Torque, 1)
275             End If
276             '
277             CAFE.MasterEngines(i).EngineNotes = GetString(j, EngineNotesColNum)
278
279             CAFE.MasterEngines(i).TechOverLUB = Me.GetTechOverRide(GetString(j, LUBColNum))
280             CAFE.MasterEngines(i).TechOverEFR = Me.GetTechOverRide(GetString(j, EFRColNum))
281             CAFE.MasterEngines(i).TechOverOHC = Me.GetTechOverRide(GetString(j, OHCColNum))
282             CAFE.MasterEngines(i).TechOverVVT = Me.GetTechOverRide(GetString(j, VVTColNum))
283             CAFE.MasterEngines(i).TechOverDISP = Me.GetTechOverRide(GetString(j,
    DISPColNum))
284             CAFE.MasterEngines(i).TechOverVVLT = Me.GetTechOverRide(GetString(j,
    VVLTColNum))
285             CAFE.MasterEngines(i).TechOverSIDI = Me.GetTechOverRide(GetString(j,
    SIDIColNum))
286             CAFE.MasterEngines(i).TechOverSUP = Me.GetTechOverRide(GetString(j, SUPColNum))
287             CAFE.MasterEngines(i).TechOverCVA = Me.GetTechOverRide(GetString(j, CVAColNum))
288             CAFE.MasterEngines(i).TechOverIVT = Me.GetTechOverRide(GetString(j, IVTColNum))
289             CAFE.MasterEngines(i).TechOverVCR = Me.GetTechOverRide(GetString(j, VCRColNum))
290             CAFE.MasterEngines(i).TechOverDSL = Me.GetTechOverRide(GetString(j, DSLcolnum))
291         Next r
292         MyBase.ValidFile = True
293     End Sub
294
295     '''
296     ' Validate string field
297     ' Simple check ...
298     ' If the field is blank, return FALSE and append string to FailureReason
299     Private Function ValidateField(ByVal field As String, _
300                     ByVal FieldName As String, _
301                     ByRef FailureReason As String) As Boolean
302         Dim Valid As Boolean = True
303         If field = String.Empty Then
304             Valid = False
305             FailureReason += FieldName + " is blank;"
306         End If
307         Return Valid
308     End Function
309
310     '''
311     ' Validate Double Field
312     ' Checks that the value is (a) a number, and (b) is between the specified min/max
313     Private Function ValidateField(ByVal field As Double, _
314                     ByVal Min As Double, ByVal Max As Double, _
315                     ByVal FieldName As String, _
316                     ByRef FailureReason As String) As Boolean
317         Dim valid = True
318         If field < Min Then
319             valid = False
320             FailureReason += FieldName + " is LT than min value of " + Min.ToString + ";"
321         ElseIf field > Max Then
322             valid = False
323             FailureReason += FieldName + " is GT than max value of " + Max.ToString + ";"
324         End If
325         Return valid
326     End Function
327
328     '''
329     ' Validate Integer Field
330     ' Checks that the value is (a) a number, and (b) is between the specified min/max
331     Private Function ValidateField(ByVal field As Integer, _
332                     ByVal Min As Integer, ByVal Max As Integer, _
333                     ByVal FieldName As String, _
334                     ByRef FailureReason As String) As Boolean
335         Dim valid = True
336         If field < Min Then
337             valid = False
```

```
338             FailureReason += FieldName + " is LT than min value of " + Min.ToString + ";"
339         ElseIf field > Max Then
340             valid = False
341             FailureReason += FieldName + " is GT than max value of " + Max.ToString + ";"
342         End If
343         Return valid
344     End Function
345
346     Private Function GetTechOverRide(ByVal TechOverRideField As Object) As CAFE.
        TechOverRide
347         Dim Str As String
348         Dim RtnCode As CAFE.TechOverRide
349         Str = GetString(TechOverRideField).ToUpper
350         If Str = "FORCE" Or Str = "F" Then
351             RtnCode = CAFE.TechOverRide.Force
352         ElseIf Str = "SKIP" Or Str = "S" Then
353             RtnCode = CAFE.TechOverRide.Skip
354         ElseIf Str = "USED" Or Str = "S" Then
355             RtnCode = CAFE.TechOverRide.Used
356         Else
357             RtnCode = CAFE.TechOverRide.None
358         End If
359         Return RtnCode
360     End Function
361
362 End Class
363
```

```
1  ' Base class for Raw input classes.
2  Public Class RawInput
3      ' stores the workbook object
4      Protected Wb As Excel.Workbook
5
6      Protected RawData(,) As Object
7      Protected RawHeaders(,) As Object
8      Protected ColMatch() As Boolean
9
10     ' stores file version info
11     Protected WbTitle As String
12     Protected WbVersion As Double
13
14     Protected ValidFile As Boolean = False  ' Set by children classes, used by IsValid()
15
16     Protected Const NewLine As String = Chr(13) + Chr(10)
17
18     ' parse workbook title and version and compare them to desired values
19     Public Function ParseWbTitle(ByVal DesiredTitle As String, ByVal DesiredVersion As
       Double) As Boolean
20         Dim title, version As String
21         Dim verIdx As Integer
22
23         title = (Wb.Title).Trim()
24
25         ' title cannot be nothing, empty, or blank
26         If IsNothing(title) OrElse title = String.Empty OrElse title = "" Then
27             ' if it is, set WbTitle and WbVersion to default values
28             Me.WbTitle = ""
29             Me.WbVersion = 0.0
30             ' return false since we need the title to proceed
31             MsgBox("The selected woorkbook is not vaiid since it " & _
32                 "does not contain workbook title and version information.", _
33                 MsgBoxStyle.Exclamation, "Missing Title/Version Information")
34             Return False
35         Else
36             ' otherwise parse title from workbook
37             verIdx = title.LastIndexOf(" ")
38             version = (title.Substring(verIdx)).Trim()
39             title = (title.Substring(0, verIdx)).Trim()
40
41             If UCase(version).StartsWith("V") Then
42                 ' title's format is: Workbook Title v.1.2
43                 ' remove "v."
44                 version = version.Substring(2)
45             Else
46                 ' title's format is: Workbook Title v 1.2
47                 '                or: Workbook Title 1.2
48                 If UCase(title).EndsWith("V") Then
49                     ' title's format is: File Title v 1.2
50                     title = (title.Substring(0, title.Length - 2)).Trim()
51                 Else
52                     ' title's format is: Workbook Title 1.2
53                     ' ** do nothing **
54                 End If
55             End If
56
57             Me.WbTitle = title
58             Me.WbVersion = GetDouble(version)
59
60             ' next, verify title and version against desired once
61             If UCase(Me.WbTitle) <> UCase(DesiredTitle) Then
62                 ' wb title does not match desired title, return false
63                 MsgBox("Selected woorkbook's title does not match desired title." & _
64                     NewLine & NewLine & "    Workbook title: """ & Me.WbTitle & """" & _
65                     NewLine & "    Desired title: """ & DesiredTitle & """" & _
66                     NewLine & NewLine & "Please select a different workbook.", _
67                     MsgBoxStyle.Exclamation, "Workbook Title Mismatch")
68                 Return False
69             End If
```

```
70
71              ' get the minimum and the maximum version that the input file is allowed to be
72              ' the min version is floor of the desired version, while the max version is
73              ' floor of the desired version + 0.9999; so if the desired version is 3.44
74              ' than the min verison is 3 and max version is 3.9999
75              Dim MinVersion As Double = Math.Floor(DesiredVersion)
76              Dim MaxVersion As Double = Math.Floor(DesiredVersion) + 0.9999
77
78              If MinVersion > Me.WbVersion OrElse Me.WbVersion > MaxVersion Then
79                  ' wb version is less than the minimum allowed version
80                  MsgBox("Selected workbook's version, " & Me.WbVersion & ", is incorrect." & ↙
    _
81                      NewLine & NewLine & "    The desired version is: " & DesiredVersion & _
82                      NewLine & "    The minimum allowed version is: " & MinVersion & _
83                      NewLine & "    The maximum allowed version is: " & MaxVersion & _
84                      NewLine & NewLine & "Please select a different workbook.", _
85                      MsgBoxStyle.Exclamation, "Workbook Version Mismatch")
86                  Return False
87              End If
88
89              ' no problems were encountered
90              Return True
91          End If
92      End Function
93
94      Public Function IsValid() As Boolean
95          Return ValidFile
96      End Function
97
98      Protected Sub CheckColumnName(ByVal DesiredTitle As String, _
99                                    ByVal rowIdx As Integer, _
100                                   ByVal colIdx As Integer, _
101                                   ByRef ColNum As Integer)
102         '
103         Me.CheckColumnName(Me.RawHeaders, DesiredTitle, rowIdx, colIdx, ColNum, True)
104     End Sub
105
106     Protected Sub CheckColumnName(ByRef RH(,) As Object, _
107                                   ByVal DesiredTitle As String, _
108                                   ByVal rowIdx As Integer, _
109                                   ByVal colIdx As Integer, _
110                                   ByRef ColNum As Integer, _
111                                   Optional ByVal SetColMatch As Boolean = False)
112         '
113         If UCase(RH(rowIdx, colIdx)) = UCase(DesiredTitle) Then
114             ColNum = colIdx
115
116             If SetColMatch Then
117                 ColMatch(colIdx) = True
118             End If
119         End If
120     End Sub
121
122     '''
123     ' Checks if a cell specified by row and col in the given RawData matches the specified
124     ' desired value.  If so, true is returned, if not, the message string (MsgString) is
125     ' appended and false is returned.
126     '
127     Protected Function VerifyRawData(ByRef RawData(,) As Object, _
128                                      ByVal DesiredValue As String, _
129                                      ByRef MsgString As String, _
130                                      ByVal row As Integer, _
131                                      ByVal col As Integer) As Boolean
132         '
133         Dim RawDataValue As String
134
135         If IsNothing(RawData(row, col)) Then
136             RawDataValue = ""
137         Else
138             RawDataValue = CType(RawData(row, col), String).Trim().ToUpper()
```

```
139         End If
140
141         If Not RawDataValue.Equals("") OrElse RawDataValue.Equals(UCase(DesiredValue)) Then
142             ' if the title in the raw data starts with the desired value,
143             ' then we have a match--return true
144             Return True
145         Else
146             ' otherwise, modify MsgString and return false
147             MsgString = MsgString + "    - " + DesiredValue + NewLine
148             Return False
149         End If
150     End Function
151
152     '''
153     ' Gets the raw data from the specified active sheet and the given ranges.
154     '
155     Protected Function GetRawData(ByVal ActSheet As Excel.Worksheet, _
156                                   ByVal Range0 As Object, _
157                                   ByVal Range1 As Object) As Object(,)
158         '
159         Return ActSheet.Range(Range0, Range1).Resize.Value
160     End Function
161
162     '''
163     ' Gets the raw data from the specified active sheet.  The returned raw data is between ↙
        the
164     ' given starting row and the last row that has data in the current active sheet.
165     '
166     Protected Function GetRawData(ByVal ActSheet As Excel.Worksheet, _
167                                   ByVal StartingRow As Integer, _
168                                   Optional ByVal MaxRow As Integer = 1024) As Object(,)
169         '
170         Dim c As Integer = 256
171         Dim r As Integer = MaxRow
172         Dim i, j As Integer
173         Dim RawData(,) As Object = Me.GetRawData(ActSheet, "A" & StartingRow, "IV" & r)
174
175         Dim incremental As Boolean = False
176         Dim lastRow As Boolean = False
177
178         ' remove empty trailing cells
179         Do
180             c -= 1
181             If c <= 0 Then Exit Do
182         Loop While RawData(1, c) Is Nothing
183
184         If c <= 0 Then Return Nothing
185
186         Do
187             If lastRow And Not incremental Then
188                 ' since we found the "last row" during geometric descent,
189                 ' there could be some rows the may have been skipped; double
190                 ' the current "last row" and switch to incremental mode
191                 r *= 2
192                 incremental = True
193                 lastRow = False
194             End If
195
196             If incremental Then
197                 r -= 1
198             Else
199                 r /= 2
200             End If
201
202             If r <= 0 Then Exit Do
203
204             For i = 0 To c
205                 ' if raw data is not nothing at the given row, then at least one column has ↙
        data...
206                 If Not IsNothing(RawData(r, c)) Then
```

```
207                     ' set lastRow to true, and exit for loop
208                     lastRow = True
209                     Exit For
210                 End If
211             Next i
212         Loop Until (lastRow And incremental)
213
214         Dim RawDataTrimed(r, c) As Object
215         For i = 1 To r
216             For j = 1 To c
217                 RawDataTrimed(i, j) = RawData(i, j)
218             Next j
219         Next i
220
221         RawDataTrimed(0, 0) = r
222         RawDataTrimed(0, 1) = c
223
224         Return RawDataTrimed
225     End Function
226
227     '''
228     ' Gets the a single row of raw data from the specified active sheet.
229     '
230     Protected Function GetRawRow(ByVal ActSheet As Excel.Worksheet, _
231                                  ByVal AtRow As Integer) As Object(,)
232         '
233         Dim RawData(,) As Object = Me.GetRawData(ActSheet, "A" & AtRow, "IV" & AtRow)
234
235         ' remove empty trailing cells
236         Dim c As Integer = 256
237         Dim r As Integer = 1
238         Do
239             c -= 1
240             If c <= 0 Then Exit Do
241         Loop While RawData(1, c) Is Nothing
242
243         If c <= 0 Then
244             Return Nothing
245         End If
246
247         Dim i As Integer
248         Dim RawDataTrimed(r, c) As Object
249         For i = 1 To c
250             RawDataTrimed(1, i) = RawData(1, i)
251         Next i
252
253         RawDataTrimed(0, 0) = r
254         RawDataTrimed(0, 1) = c
255
256         Return RawDataTrimed
257     End Function
258
259     ' checks if the given column number is 0 and appends the message string,
260     ' MsgString, with the provided error message, ErrorMsg, if so.
261     Public Function CheckRequiredColumn(ByVal ColNum As Integer, _
262                                         ByVal ErrorMsg As String, _
263                                         ByRef MsgString As String) As Boolean
264         '
265         If ColNum = 0 Then
266             MsgString += (ErrorMsg & NewLine)
267             Return False
268         Else
269             Return True
270         End If
271     End Function
272
273     ' Get a double value from RawData(r,c)
274     Public Function GetDouble(ByVal r As Integer, ByVal c As Integer) As Double
275         Return GetDouble(RawData, r, c)
276     End Function
```

```
277     ' Get a double value from a 2d Array of Objects
278     Public Function GetDouble(ByRef ObjArray2d(,) As Object, ByVal r As Integer, ByVal c As Integer) As Double
279         If r <= 0 OrElse c <= 0 Then
280             Return 0.0
281         Else
282             Return GetDouble(ObjArray2d(r, c))
283         End If
284     End Function
285     ' Get a double value from a 1d array of objects
286     Public Function GetDouble(ByRef ObjArray() As Object, ByVal idx As Integer) As Double
287         Return GetDouble(ObjArray(idx))
288     End Function
289     ' Get a double value from an object
290     Public Function GetDouble(ByVal v As Object) As Double
291         Dim DblV As Double ' auto initializes to 0.0
292         If IsNothing(v) = False Then
293             If IsNumeric(v) Then
294                 DblV = CDbl(v)
295             End If
296         End If
297         Return DblV
298     End Function
299
300     ' Get an integer value from RawData(r,c)
301     Public Function GetInteger(ByVal r As Integer, ByVal c As Integer) As Integer
302         Return GetInteger(RawData, r, c)
303     End Function
304     ' Get an integer value from a 2d array of objects
305     Public Function GetInteger(ByRef ObjArray2d(,) As Object, ByVal r As Integer, ByVal c As Integer) As Integer
306         If r <= 0 OrElse c <= 0 Then
307             Return 0
308         Else
309             Return GetInteger(ObjArray2d(r, c))
310         End If
311     End Function
312     ' Get an integer value from a 1d array of objects
313     Public Function GetInteger(ByRef ObjArray() As Object, ByVal idx As Integer) As Integer
314         Return GetInteger(ObjArray(idx))
315     End Function
316     ' Get an integer value from an object
317     Public Function GetInteger(ByVal v As Object) As Integer
318         Dim IntV As Integer ' auto initializes to 0
319         If IsNothing(v) = False Then
320             If IsNumeric(v) Then
321                 IntV = CInt(v)
322             End If
323         End If
324         GetInteger = IntV
325     End Function
326
327     ' Get a string from RawData(r,c)
328     Public Function GetString(ByVal r As Integer, ByVal c As Integer) As String
329         Return GetString(RawData, r, c)
330     End Function
331     ' Get a string value from a 2d array of objects
332     Public Function GetString(ByRef ObjArray2d(,) As Object, ByVal r As Integer, ByVal c As Integer) As String
333         If r <= 0 OrElse c <= 0 Then
334             Return ""
335         Else
336             Return GetString(ObjArray2d(r, c))
337         End If
338     End Function
339     ' Get a string value from an array ob objects
340     Public Function GetString(ByRef ObjArray() As Object, ByVal idx As Integer) As String
341         Return GetString(ObjArray(idx))
342     End Function
343     ' Get a string from an Object
```

```
344     Public Function GetString(ByVal v As Object) As String
345         Dim StrV As String = String.Empty
346         If IsNothing(v) = False Then
347             StrV = CStr(v)
348         End If
349         GetString = StrV
350     End Function
351
352     ' Get a boolean from RawData(r,c)
353     Public Function GetBoolean(ByVal r As Integer, ByVal c As Integer) As Boolean
354         Return GetBoolean(RawData, r, c)
355     End Function
356     ' Get a boolean from a 2d Array of Objects
357     Public Function GetBoolean(ByRef ObjArray2d(,) As Object, ByVal r As Integer, ByVal c
        As Integer) As Boolean
358         If r <= 0 OrElse c <= 0 Then
359             Return False
360         Else
361             Return GetBoolean(ObjArray2d(r, c))
362         End If
363     End Function
364     ' Get a boolean from an array of objects
365     Public Function GetBoolean(ByRef ObjArray As Object, ByVal idx As Integer) As Boolean
366         Return GetBoolean(ObjArray(idx))
367     End Function
368     ' Get a boolean from an object
369     Public Function GetBoolean(ByVal v As Object) As Boolean
370         Dim BoolV As Boolean
371         Dim BString As String = GetString(v).ToUpper
372         If (String.Compare(BString, "TRUE") = 0) Then
373             BoolV = True        ' If it's "TRUE" then it's true.
374         Else
375             BoolV = False       ' EVERTHING else is false.
376         End If
377         Return BoolV
378     End Function
379
380 End Class
381
```

```
1  Public Class RawMarketDataClass
2      Inherits RawInput
3
4      Protected Sub New()
5      End Sub
6
7      Public Sub New(ByVal fName As String, _
8                     ByRef RawVehicles As RawVehiclesClass, _
9                     ByRef RawEngines As RawEnginesClass, _
10                    ByRef RawTransmissions As RawTransmissionsClass)
11         '
12         ' read market data file and store raw data
13         Dim MarketDataFile As New C_Cafe_InFile
14         Dim RowCount As Integer
15         Dim ColCount As Integer
16         Dim ActSheet As Excel.Worksheet
17         Dim SheetName As String
18         Dim s As Integer
19
20         If Not MarketDataFile.OpenFile(fName, 3) Then
21             ' could not open file for any reason, prompt user and exit...
22             MsgBox("An error has occured while opening the market data input file.  " & _
23                    "Please try again.", MsgBoxStyle.Exclamation, "Error Opening File")
24             Exit Sub
25         End If
26
27         Wb = MarketDataFile.GetWorkbookObject
28
29         ' parse wb title
30         If Not Me.ParseWbTitle("CAFE Market Data Dataset file", 1.0) Then
31             ' if wb title or version did not match desired ones
32             ' close file and exit sub
33             MarketDataFile.CloseFile()
34             MarketDataFile = Nothing
35             Exit Sub
36         End If
37
38         ' scan each worksheet for vehicles, engines, and transmissions worksheets
39
40         For s = 1 To Wb.Worksheets.Count
41             ActSheet = MarketDataFile.SetSheet(s)
42             SheetName = ActSheet.Name
43
44             RowCount = MarketDataFile.GetNumRows(ActSheet)
45             ColCount = MarketDataFile.GetNumColumns(ActSheet, 1, UCase(SheetName) =
    "VEHICLES")
46
47             Select Case UCase(SheetName)
48                 Case "VEHICLES"
49                     ' use VehiclesClass
50                     RawVehicles = New RawVehiclesClass
51                     RawVehicles.LoadData(ActSheet, RowCount, ColCount)
52
53                 Case "ENGINES"
54                     ' use EnginesClass
55                     RawEngines = New RawEnginesClass
56                     RawEngines.LoadData(ActSheet, RowCount, ColCount)
57
58                 Case "TRANSMISSIONS"
59                     ' use TransmissionsClass
60                     RawTransmissions = New RawTransmissionsClass
61                     RawTransmissions.LoadData(ActSheet, RowCount, ColCount)
62             End Select
63         Next s
64
65         ' Close the file
66         MarketDataFile.CloseFile()
67         MarketDataFile = Nothing
68
69         Me.ValidFile = True
```

```
70      End Sub
71
72  End Class
73
```

```
Public Class RawParametersClass
    Inherits RawInput

 Data


    '
    ' constructors
    '
    Public Sub New()
    End Sub

    Public Sub New(ByVal fName As String)
        ' get parameters file
        Me.ParametersFile = New C_Cafe_InFile
        Me.fName = fName

        Dim sRange As Excel.Range
        Dim eRange As Excel.Range

        Dim i, j As Integer

        ' stores the error msg to display to user in case of error
        Dim MsgString As String = ""

        If Not Me.ParametersFile.OpenFile(fName) Then
            ' could not open file for any reason, prompt user and exit...
            MsgBox("An error has occured while opening the parameters input file.  " & _
                   "Please try again.", MsgBoxStyle.Exclamation, "Error Opening File")
            Exit Sub
        End If

        Me.Wb = ParametersFile.GetWorkbookObject()

        ' parse wb title
        If Not Me.ParseWbTitle("CAFE Parameters file", 3.0) Then
            ' if wb title or version did not match desired once
            ' close file and exit sub
            ParametersFile.CloseFile()
            ParametersFile = Nothing
            Exit Sub
        End If

        ' workbook title and version matched, continue...
        ParametersFile.Activate()

        ' load worksheets
        If Not Me.LoadWorksheets() Then
            MsgBox("Failed to load file because of missing data.", _
                   MsgBoxStyle.Exclamation, "Missing data.")
            Me.CloseFile()
            Exit Sub
        End If

        ' create new instance of ParametersClass
        Me.Parameters = New ParametersClass

        ' since all worksheets were loaded successfully, get number of mfgs and mys
        Me.GetNumMfgAndMY()

        ' next, get raw data
        Me.LoadRawData(Me.GeneralParameters, Me.RawGeneralParameters, 2, 1, 6, 2)
        Me.LoadRawData(Me.CostAllocation, Me.RawCostAllocation, 3, 1, 3 + Me.nMfg, 2)
        Me.LoadRawData(Me.MY02LTWeight, Me.RawMY02LTWeight, 3, 1, 3 + Me.nMfg, 2)
        Me.LoadRawData(Me.WillingToPayFines, Me.RawWillingToPayFines, 3, 1, 3 + Me.nMfg, 1 + Me.nMY)
        If Me.NMNLIncluded Then
            ' cannot load nml worksheet if it is not inculded
            Me.LoadRawData(Me.NMNL, Me.RawNMNL, 3, 1, 17, 3)
```

```
69          End If
70
71          ' verify compliance data...
72          MsgString = ""
73
74          ' verify general parameters
75          Me.VerifyRawData(Me.RawGeneralParameters, "Discount Rate", MsgString, 1, 1)
76          Me.VerifyRawData(Me.RawGeneralParameters, "Payback Period", MsgString, 2, 1)
77          Me.VerifyRawData(Me.RawGeneralParameters, "FE Shortfall", MsgString, 3, 1)
78          Me.VerifyRawData(Me.RawGeneralParameters, "Kf", MsgString, 4, 1)
79          Me.VerifyRawData(Me.RawGeneralParameters, "kCurbWeightDelta", MsgString, 5, 1)
80
81          ' verify cost allocation
82          Me.VerifyRawData(Me.RawCostAllocation, "Mfr. Code", MsgString, 1, 1)
83
84          ' verify my02ltweight
85          Me.VerifyRawData(Me.RawMY02LTWeight, "Mfr. Code", MsgString, 1, 1)
86
87          ' verify willing to pay fines
88          Me.VerifyRawData(Me.RawWillingToPayFines, "Mfr. Code", MsgString, 1, 1)
89
90          ' check if there are any errors
91          If MsgString <> "" Then
92              ' one or more of the params is not found
93              MsgBox("The following parameter section(s) were not found: " & _
94                     NewLine & NewLine & MsgString & NewLine & "Possibly the input " & _
95                     "parameters file (" & fName & ") is not properly alligned.", _
96                     MsgBoxStyle.Exclamation, "Missing Parameters Information")
97              ' exit the sub without loading any data
98              Me.CloseFile()
99              Exit Sub
100         End If
101
102         ' verify nmnl only if the worksheet is present
103         If Me.NMNLIncluded Then
104             MsgString = ""      ' reset msg
105
106             Me.VerifyRawData(Me.RawNMNL, "Inclusive Value Parameter", MsgString, 1, 1)
107             Me.VerifyRawData(Me.RawNMNL, "Attribute", MsgString, 3, 1)
108             Me.VerifyRawData(Me.RawNMNL, "Units", MsgString, 3, 2)
109             Me.VerifyRawData(Me.RawNMNL, "Coefficient", MsgString, 3, 3)
110             Me.VerifyRawData(Me.RawNMNL, "Number of Market Segments", MsgString, 15, 1)
111
112             ' check if there were any errors verifing NML sheet
113             If MsgString <> "" Then
114                 Me.NMNLIncluded = False
115                 CAFE.NMNLIncluded = False
116             End If
117         End If
118
119         ' load general parameters
120         'Me.Parameters.discountRate = Me.GetDouble(Me.GeneralParameters, 1, 2)
121         Me.Parameters.Payback = Me.GetDouble(Me.RawGeneralParameters, 2, 2)
122         Me.Parameters.FE_Shortfall = Me.GetDouble(Me.RawGeneralParameters, 3, 2)
123         Me.Parameters.K_f = Me.GetDouble(Me.RawGeneralParameters, 4, 2)
124         Me.Parameters.kCurbWeightDelta = Me.GetDouble(Me.RawGeneralParameters, 5, 2)
125
126         ' load cost allocation
127         ReDim Me.Parameters.CostAllocationStrategy(Me.nMfg - 1)
128
129         For i = 0 To Me.nMfg - 1
130             Me.Parameters.CostAllocationStrategy(i) = GetInteger(Me.RawCostAllocation, 2 + i, ↙
        2)
131         Next i
132
133         ' load my02ltweight
134         ReDim Me.Parameters.MY02LTWeight(Me.nMfg - 1)
135
136         For i = 0 To Me.nMfg - 1
```

```
137             Me.Parameters.MY02LTWeight(i) = Me.GetInteger(Me.RawMY02LTWeight, 2 + i, 2)
138         Next i
139
140         ' load willing to pay fines
141         ReDim Me.Parameters.MfgIDs(Me.nMfg - 1)
142         ReDim Me.Parameters.WillingToPayFines(CAFE.GMYI(Me.Parameters.wtpfMaxYear, False), Me.nMfg - 1)
143         ReDim Me.Parameters.WillingToPayFinesMfgList(Me.nMfg - 1)
144
145         For i = 0 To Me.nMfg - 1
146             ' first get the manufacturer id (name)
147             Me.Parameters.MfgIDs(i) = GetString(Me.RawWillingToPayFines, 2 + i, 1)
148             Me.Parameters.WillingToPayFinesMfgList(i) = Me.Parameters.MfgIDs(i)
149         Next i
150
151         For i = 0 To Me.nMY - 1
152             Dim m As Integer = CAFE.GMYI(Me.Parameters.wtpfMY(i), False)
153             ' then get "manufacturer willing to pay"
154             For j = 0 To Me.nMfg - 1
155                 Dim str As String = GetString(Me.RawWillingToPayFines, 2 + j, 2 + i)
156                 Me.Parameters.WillingToPayFines(m, j) = (UCase(str) = "Y")
157             Next j
158         Next i
159
160         ' load nmnl
161         If Me.NMNLIncluded Then
162             ReDim Me.Parameters.NMNLCoeff(10)
163
164             Me.Parameters.NMNLCoeff(0) = GetDouble(Me.RawNMNL, 1, 3)
165             ' Then get coefficients for different vehicle attributes.
166             For i = 1 To 10
167                 Me.Parameters.NMNLCoeff(i) = GetDouble(Me.RawNMNL, i + 3, 3)
168             Next i
169             ' Finally, get number of market segments
170             Me.Parameters.nMktSegments = Int(GetDouble(Me.RawNMNL, 15, 3))
171         End If
172
173         ' assign values to arrays in vehicleAgeData worksheet
174         ' vehicle age of 0 will be found in array(0,0)
175         Dim f As Integer
176         Dim g As Integer
177         g = 0
178         For f = 0 To 25
179             Me.Parameters.milesAuto(f, g) = vehicleAgeData.Range("R" & f + 5).Value()
180             Me.Parameters.milesPickup(f, g) = vehicleAgeData.Range("S" & f + 5).Value()
181             Me.Parameters.milesVan(f, g) = vehicleAgeData.Range("T" & f + 5).Value()
182             Me.Parameters.milesSuv(f, g) = vehicleAgeData.Range("U" & f + 5).Value()
183             Me.Parameters.milesLTDefault(f, g) = vehicleAgeData.Range("V" & f + 5).Value()
184
185             ''Me.Parameters.milesLDGV(f, g) = vehicleAgeData.Range("J" & f + 5).Value()
186             ''Me.Parameters.milesLDDV(f, g) = vehicleAgeData.Range("K" & f + 5).Value()
187             ''Me.Parameters.milesLDGT1(f, g) = vehicleAgeData.Range("L" & f + 5).Value()
188             ''Me.Parameters.milesLDGT2(f, g) = vehicleAgeData.Range("M" & f + 5).Value()
189             ''Me.Parameters.milesLDDV1(f, g) = vehicleAgeData.Range("N" & f + 5).Value()
190             ''Me.Parameters.milesLDDV2(f, g) = vehicleAgeData.Range("O" & f + 5).Value()
191             ''Me.Parameters.milesHDGV2b(f, g) = vehicleAgeData.Range("P" & f + 5).Value()
192             ''Me.Parameters.milesHDDV2b(f, g) = vehicleAgeData.Range("Q" & f + 5).Value()
193
194             Me.Parameters.survivalLDGV(f, g) = vehicleAgeData.Range("B" & f + 5).Value()
195             Me.Parameters.survivalLDDV(f, g) = vehicleAgeData.Range("C" & f + 5).Value()
196             Me.Parameters.survivalLDGT1(f, g) = vehicleAgeData.Range("D" & f + 5).Value()
197             Me.Parameters.survivalLDGT2(f, g) = vehicleAgeData.Range("E" & f + 5).Value()
198             Me.Parameters.survivalLDDV1(f, g) = vehicleAgeData.Range("F" & f + 5).Value()
199             Me.Parameters.survivalLDDV2(f, g) = vehicleAgeData.Range("G" & f + 5).Value()
200             Me.Parameters.survivalHDGV2b(f, g) = vehicleAgeData.Range("H" & f + 5).Value()
201             Me.Parameters.survivalHDDV2b(f, g) = vehicleAgeData.Range("I" & f + 5).Value()
202         Next f
203
204         'assign values to vars in fuelProperties worksheet
```

```
205         Me.Parameters.convGasEngDens = fuelProperties.Range("B5").Value()
206         Me.Parameters.fedGasEngDens = fuelProperties.Range("B6").Value()
207         Me.Parameters.caliGasEngDens = fuelProperties.Range("B7").Value()
208         Me.Parameters.convDiesEngDens = fuelProperties.Range("B8").Value()
209         Me.Parameters.lowDiesEngDens = fuelProperties.Range("B9").Value()
210         Me.Parameters.convGasMassDens = fuelProperties.Range("C5").Value()
211         Me.Parameters.fedGasMassDens = fuelProperties.Range("C6").Value()
212         Me.Parameters.caliGasMassDens = fuelProperties.Range("C7").Value()
213         Me.Parameters.convDiesMassDens = fuelProperties.Range("C8").Value()
214         Me.Parameters.lowDiesMassDens = fuelProperties.Range("C9").Value()
215         Me.Parameters.convGasCarb = fuelProperties.Range("D5").Value()
216         Me.Parameters.fedGasCarb = fuelProperties.Range("D6").Value()
217         Me.Parameters.caliGasCarb = fuelProperties.Range("D7").Value()
218         Me.Parameters.convDiesCarb = fuelProperties.Range("D8").Value()
219         Me.Parameters.lowDiesCarb = fuelProperties.Range("D9").Value()
220         Me.Parameters.convGasSulf = fuelProperties.Range("E5").Value()
221         Me.Parameters.fedGasSulf = fuelProperties.Range("E6").Value()
222         Me.Parameters.caliGasSulf = fuelProperties.Range("E7").Value()
223         Me.Parameters.convDiesSulf = fuelProperties.Range("E8").Value()
224         Me.Parameters.lowDiesSulf = fuelProperties.Range("E9").Value()
225         Me.Parameters.gasShareImp = fuelProperties.Range("B19").Value()
226         Me.Parameters.gasShareDomRef = fuelProperties.Range("B21").Value()
227         Me.Parameters.gasShareDomRefDomCrude = fuelProperties.Range("B23").Value()
228         Me.Parameters.gasShareDomRefImpCrude = fuelProperties.Range("B24").Value()
229         Me.Parameters.gasLowImp = fuelProperties.Range("B25").Value()
230         Me.Parameters.gasLowDomRef = fuelProperties.Range("B26").Value()
231         Me.Parameters.gasLowDomRefDomCrude = fuelProperties.Range("B27").Value()
232         Me.Parameters.gasLowDomRefImpCrude = fuelProperties.Range("B29").Value()
233         Me.Parameters.diesShareImp = fuelProperties.Range("C19").Value()
234         Me.Parameters.diesShareDomRef = fuelProperties.Range("C21").Value()
235         Me.Parameters.diesShareDomRefDomCrude = fuelProperties.Range("C23").Value()
236         Me.Parameters.diesShareDomRefImpCrude = fuelProperties.Range("C24").Value()
237         Me.Parameters.diesLowImp = fuelProperties.Range("C25").Value()
238         Me.Parameters.diesLowDomRef = fuelProperties.Range("C26").Value()
239         Me.Parameters.diesLowDomRefDomCrude = fuelProperties.Range("C27").Value()
240         Me.Parameters.diesLowDomRefImpCrude = fuelProperties.Range("C29").Value()
241         'assumed fuel mix
242         Me.Parameters.convGasMix = fuelProperties.Range("B39").Value()
243         Me.Parameters.fedGasMix = fuelProperties.Range("B40").Value()
244         Me.Parameters.fedNorthGasMix = fuelProperties.Range("B41").Value()
245         Me.Parameters.fedSouthGasMix = fuelProperties.Range("B42").Value()
246         Me.Parameters.caliGasMix = fuelProperties.Range("B43").Value()
247         Me.Parameters.convDiesMix = fuelProperties.Range("B44").Value()
248         Me.Parameters.lowDiesMix = fuelProperties.Range("B45").Value()
249
250         ' assign values to upstreamEmissions vars
251         Me.Parameters.totExtrConvGasCO = upstreamEmissions.Range("C7").Value()
252         Me.Parameters.totExtrFedGasCO = upstreamEmissions.Range("C8").Value()
253         Me.Parameters.totExtrCaliGasCO = upstreamEmissions.Range("C9").Value()
254         Me.Parameters.totExtrConvDiesCO = upstreamEmissions.Range("C10").Value()
255         Me.Parameters.totExtrLowDiesCO = upstreamEmissions.Range("C11").Value()
256         Me.Parameters.totExtrConvGasVOC = upstreamEmissions.Range("C12").Value()
257         Me.Parameters.totExtrFedGasVOC = upstreamEmissions.Range("C13").Value()
258         Me.Parameters.totExtrCaliGasVOC = upstreamEmissions.Range("C14").Value()
259         Me.Parameters.totExtrConvDiesVOC = upstreamEmissions.Range("C15").Value()
260         Me.Parameters.totExtrLowDiesVOC = upstreamEmissions.Range("C16").Value()
261         Me.Parameters.totExtrConvGasNOX = upstreamEmissions.Range("C17").Value()
262         Me.Parameters.totExtrFedGasNOX = upstreamEmissions.Range("C18").Value()
263         Me.Parameters.totExtrCaliGasNOX = upstreamEmissions.Range("C19").Value()
264         Me.Parameters.totExtrConvDiesNOX = upstreamEmissions.Range("C20").Value()
265         Me.Parameters.totExtrLowDiesNOX = upstreamEmissions.Range("C21").Value()
266         Me.Parameters.totExtrConvGasSOX = upstreamEmissions.Range("C22").Value()
267         Me.Parameters.totExtrFedGasSOX = upstreamEmissions.Range("C23").Value()
268         Me.Parameters.totExtrCaliGasSOX = upstreamEmissions.Range("C24").Value()
269         Me.Parameters.totExtrConvDiesSOX = upstreamEmissions.Range("C25").Value()
270         Me.Parameters.totExtrLowDiesSOX = upstreamEmissions.Range("C26").Value()
271         Me.Parameters.totExtrConvGasPM = upstreamEmissions.Range("C27").Value()
272         Me.Parameters.totExtrFedGasPM = upstreamEmissions.Range("C28").Value()
273         Me.Parameters.totExtrCaliGasPM = upstreamEmissions.Range("C29").Value()
```

```
274         Me.Parameters.totExtrConvDiesPM = upstreamEmissions.Range("C30").Value()
275         Me.Parameters.totExtrLowDiesPM = upstreamEmissions.Range("C31").Value()
276         Me.Parameters.totExtrConvGasCO2 = upstreamEmissions.Range("C32").Value()
277         Me.Parameters.totExtrFedGasCO2 = upstreamEmissions.Range("C33").Value()
278         Me.Parameters.totExtrCaliGasCO2 = upstreamEmissions.Range("C34").Value()
279         Me.Parameters.totExtrConvDiesCO2 = upstreamEmissions.Range("C35").Value()
280         Me.Parameters.totExtrLowDiesCO2 = upstreamEmissions.Range("C36").Value()
281         Me.Parameters.totExtrConvGasGHG = upstreamEmissions.Range("C37").Value()
282         Me.Parameters.totExtrFedGasGHG = upstreamEmissions.Range("C38").Value()
283         Me.Parameters.totExtrCaliGasGHG = upstreamEmissions.Range("C39").Value()
284         Me.Parameters.totExtrConvDiesGHG = upstreamEmissions.Range("C40").Value()
285         Me.Parameters.totExtrLowDiesGHG = upstreamEmissions.Range("C41").Value()
286         Me.Parameters.totTransConvGasCO = upstreamEmissions.Range("D7").Value()
287         Me.Parameters.totTransFedGasCO = upstreamEmissions.Range("D8").Value()
288         Me.Parameters.totTransCaliGasCO = upstreamEmissions.Range("D9").Value()
289         Me.Parameters.totTransConvDiesCO = upstreamEmissions.Range("D10").Value()
290         Me.Parameters.totTransLowDiesCO = upstreamEmissions.Range("D11").Value()
291         Me.Parameters.totTransConvGasVOC = upstreamEmissions.Range("D12").Value()
292         Me.Parameters.totTransFedGasVOC = upstreamEmissions.Range("D13").Value()
293         Me.Parameters.totTransCaliGasVOC = upstreamEmissions.Range("D14").Value()
294         Me.Parameters.totTransConvDiesVOC = upstreamEmissions.Range("D15").Value()
295         Me.Parameters.totTransLowDiesVOC = upstreamEmissions.Range("D16").Value()
296         Me.Parameters.totTransConvGasNOX = upstreamEmissions.Range("D17").Value()
297         Me.Parameters.totTransFedGasNOX = upstreamEmissions.Range("D18").Value()
298         Me.Parameters.totTransCaliGasNOX = upstreamEmissions.Range("D19").Value()
299         Me.Parameters.totTransConvDiesNOX = upstreamEmissions.Range("D20").Value()
300         Me.Parameters.totTransLowDiesNOX = upstreamEmissions.Range("D21").Value()
301         Me.Parameters.totTransConvGasSOX = upstreamEmissions.Range("D22").Value()
302         Me.Parameters.totTransFedGasSOX = upstreamEmissions.Range("D23").Value()
303         Me.Parameters.totTransCaliGasSOX = upstreamEmissions.Range("D24").Value()
304         Me.Parameters.totTransConvDiesSOX = upstreamEmissions.Range("D25").Value()
305         Me.Parameters.totTransLowDiesSOX = upstreamEmissions.Range("D26").Value()
306         Me.Parameters.totTransConvGasPM = upstreamEmissions.Range("D27").Value()
307         Me.Parameters.totTransFedGasPM = upstreamEmissions.Range("D28").Value()
308         Me.Parameters.totTransCaliGasPM = upstreamEmissions.Range("D29").Value()
309         Me.Parameters.totTransConvDiesPM = upstreamEmissions.Range("D30").Value()
310         Me.Parameters.totTransLowDiesPM = upstreamEmissions.Range("D31").Value()
311         Me.Parameters.totTransConvGasCO2 = upstreamEmissions.Range("D32").Value()
312         Me.Parameters.totTransFedGasCO2 = upstreamEmissions.Range("D33").Value()
313         Me.Parameters.totTransCaliGasCO2 = upstreamEmissions.Range("D34").Value()
314         Me.Parameters.totTransConvDiesCO2 = upstreamEmissions.Range("D35").Value()
315         Me.Parameters.totTransLowDiesCO2 = upstreamEmissions.Range("D36").Value()
316         Me.Parameters.totTransConvGasGHG = upstreamEmissions.Range("D37").Value()
317         Me.Parameters.totTransFedGasGHG = upstreamEmissions.Range("D38").Value()
318         Me.Parameters.totTransCaliGasGHG = upstreamEmissions.Range("D39").Value()
319         Me.Parameters.totTransConvDiesGHG = upstreamEmissions.Range("D40").Value()
320         Me.Parameters.totTransLowDiesGHG = upstreamEmissions.Range("D41").Value()
321         Me.Parameters.totRefConvGasCO = upstreamEmissions.Range("E7").Value()
322         Me.Parameters.totRefFedGasCO = upstreamEmissions.Range("E8").Value()
323         Me.Parameters.totRefCaliGasCO = upstreamEmissions.Range("E9").Value()
324         Me.Parameters.totRefConvDiesCO = upstreamEmissions.Range("E10").Value()
325         Me.Parameters.totRefLowDiesCO = upstreamEmissions.Range("E11").Value()
326         Me.Parameters.totRefConvGasVOC = upstreamEmissions.Range("E12").Value()
327         Me.Parameters.totRefFedGasVOC = upstreamEmissions.Range("E13").Value()
328         Me.Parameters.totRefCaliGasVOC = upstreamEmissions.Range("E14").Value()
329         Me.Parameters.totRefConvDiesVOC = upstreamEmissions.Range("E15").Value()
330         Me.Parameters.totRefLowDiesVOC = upstreamEmissions.Range("E16").Value()
331         Me.Parameters.totRefConvGasNOX = upstreamEmissions.Range("E17").Value()
332         Me.Parameters.totRefFedGasNOX = upstreamEmissions.Range("E18").Value()
333         Me.Parameters.totRefCaliGasNOX = upstreamEmissions.Range("E19").Value()
334         Me.Parameters.totRefConvDiesNOX = upstreamEmissions.Range("E20").Value()
335         Me.Parameters.totRefLowDiesNOX = upstreamEmissions.Range("E21").Value()
336         Me.Parameters.totRefConvGasSOX = upstreamEmissions.Range("E22").Value()
337         Me.Parameters.totRefFedGasSOX = upstreamEmissions.Range("E23").Value()
338         Me.Parameters.totRefCaliGasSOX = upstreamEmissions.Range("E24").Value()
339         Me.Parameters.totRefConvDiesSOX = upstreamEmissions.Range("E25").Value()
340         Me.Parameters.totRefLowDiesSOX = upstreamEmissions.Range("E26").Value()
341         Me.Parameters.totRefConvGasPM = upstreamEmissions.Range("E27").Value()
342         Me.Parameters.totRefFedGasPM = upstreamEmissions.Range("E28").Value()
```

```
343         Me.Parameters.totRefCaliGasPM = upstreamEmissions.Range("E29").Value()
344         Me.Parameters.totRefConvDiesPM = upstreamEmissions.Range("E30").Value()
345         Me.Parameters.totRefLowDiesPM = upstreamEmissions.Range("E31").Value()
346         Me.Parameters.totRefConvGasCO2 = upstreamEmissions.Range("E32").Value()
347         Me.Parameters.totRefFedGasCO2 = upstreamEmissions.Range("E33").Value()
348         Me.Parameters.totRefCaliGasCO2 = upstreamEmissions.Range("E34").Value()
349         Me.Parameters.totRefConvDiesCO2 = upstreamEmissions.Range("E35").Value()
350         Me.Parameters.totRefLowDiesCO2 = upstreamEmissions.Range("E36").Value()
351         Me.Parameters.totRefConvGasGHG = upstreamEmissions.Range("E37").Value()
352         Me.Parameters.totRefFedGasGHG = upstreamEmissions.Range("E38").Value()
353         Me.Parameters.totRefCaliGasGHG = upstreamEmissions.Range("E39").Value()
354         Me.Parameters.totRefConvDiesGHG = upstreamEmissions.Range("E40").Value()
355         Me.Parameters.totRefLowDiesGHG = upstreamEmissions.Range("E41").Value()
356         Me.Parameters.totTSDConvGasCO = upstreamEmissions.Range("F7").Value()
357         Me.Parameters.totTSDFedGasCO = upstreamEmissions.Range("F8").Value()
358         Me.Parameters.totTSDCaliGasCO = upstreamEmissions.Range("F9").Value()
359         Me.Parameters.totTSDConvDiesCO = upstreamEmissions.Range("F10").Value()
360         Me.Parameters.totTSDLowDiesCO = upstreamEmissions.Range("F11").Value()
361         Me.Parameters.totTSDConvGasVOC = upstreamEmissions.Range("F12").Value()
362         Me.Parameters.totTSDFedGasVOC = upstreamEmissions.Range("F13").Value()
363         Me.Parameters.totTSDCaliGasVOC = upstreamEmissions.Range("F14").Value()
364         Me.Parameters.totTSDConvDiesVOC = upstreamEmissions.Range("F15").Value()
365         Me.Parameters.totTSDLowDiesVOC = upstreamEmissions.Range("F16").Value()
366         Me.Parameters.totTSDConvGasNOX = upstreamEmissions.Range("F17").Value()
367         Me.Parameters.totTSDFedGasNOX = upstreamEmissions.Range("F18").Value()
368         Me.Parameters.totTSDCaliGasNOX = upstreamEmissions.Range("F19").Value()
369         Me.Parameters.totTSDConvDiesNOX = upstreamEmissions.Range("F20").Value()
370         Me.Parameters.totTSDLowDiesNOX = upstreamEmissions.Range("F21").Value()
371         Me.Parameters.totTSDConvGasSOX = upstreamEmissions.Range("F22").Value()
372         Me.Parameters.totTSDFedGasSOX = upstreamEmissions.Range("F23").Value()
373         Me.Parameters.totTSDCaliGasSOX = upstreamEmissions.Range("F24").Value()
374         Me.Parameters.totTSDConvDiesSOX = upstreamEmissions.Range("F25").Value()
375         Me.Parameters.totTSDLowDiesSOX = upstreamEmissions.Range("F26").Value()
376         Me.Parameters.totTSDConvGasPM = upstreamEmissions.Range("F27").Value()
377         Me.Parameters.totTSDFedGasPM = upstreamEmissions.Range("F28").Value()
378         Me.Parameters.totTSDCaliGasPM = upstreamEmissions.Range("F29").Value()
379         Me.Parameters.totTSDConvDiesPM = upstreamEmissions.Range("F30").Value()
380         Me.Parameters.totTSDLowDiesPM = upstreamEmissions.Range("F31").Value()
381         Me.Parameters.totTSDConvGasCO2 = upstreamEmissions.Range("F32").Value()
382         Me.Parameters.totTSDFedGasCO2 = upstreamEmissions.Range("F33").Value()
383         Me.Parameters.totTSDCaliGasCO2 = upstreamEmissions.Range("F34").Value()
384         Me.Parameters.totTSDConvDiesCO2 = upstreamEmissions.Range("F35").Value()
385         Me.Parameters.totTSDLowDiesCO2 = upstreamEmissions.Range("F36").Value()
386         Me.Parameters.totTSDConvGasGHG = upstreamEmissions.Range("F37").Value()
387         Me.Parameters.totTSDFedGasGHG = upstreamEmissions.Range("F38").Value()
388         Me.Parameters.totTSDCaliGasGHG = upstreamEmissions.Range("F39").Value()
389         Me.Parameters.totTSDConvDiesGHG = upstreamEmissions.Range("F40").Value()
390         Me.Parameters.totTSDLowDiesGHG = upstreamEmissions.Range("F41").Value()
391         Me.Parameters.urbExtrConvGasCO = upstreamEmissions.Range("G7").Value()
392         Me.Parameters.urbExtrFedGasCO = upstreamEmissions.Range("G8").Value()
393         Me.Parameters.urbExtrCaliGasCO = upstreamEmissions.Range("G9").Value()
394         Me.Parameters.urbExtrConvDiesCO = upstreamEmissions.Range("G10").Value()
395         Me.Parameters.urbExtrLowDiesCO = upstreamEmissions.Range("G11").Value()
396         Me.Parameters.urbExtrConvGasVOC = upstreamEmissions.Range("G12").Value()
397         Me.Parameters.urbExtrFedGasVOC = upstreamEmissions.Range("G13").Value()
398         Me.Parameters.urbExtrCaliGasVOC = upstreamEmissions.Range("G14").Value()
399         Me.Parameters.urbExtrConvDiesVOC = upstreamEmissions.Range("G15").Value()
400         Me.Parameters.urbExtrLowDiesVOC = upstreamEmissions.Range("G16").Value()
401         Me.Parameters.urbExtrConvGasNOX = upstreamEmissions.Range("G17").Value()
402         Me.Parameters.urbExtrFedGasNOX = upstreamEmissions.Range("G18").Value()
403         Me.Parameters.urbExtrCaliGasNOX = upstreamEmissions.Range("G19").Value()
404         Me.Parameters.urbExtrConvDiesNOX = upstreamEmissions.Range("G20").Value()
405         Me.Parameters.urbExtrLowDiesNOX = upstreamEmissions.Range("G21").Value()
406         Me.Parameters.urbExtrConvGasSOX = upstreamEmissions.Range("G22").Value()
407         Me.Parameters.urbExtrFedGasSOX = upstreamEmissions.Range("G23").Value()
408         Me.Parameters.urbExtrCaliGasSOX = upstreamEmissions.Range("G24").Value()
409         Me.Parameters.urbExtrConvDiesSOX = upstreamEmissions.Range("G25").Value()
410         Me.Parameters.urbExtrLowDiesSOX = upstreamEmissions.Range("G26").Value()
411         Me.Parameters.urbExtrConvGasPM = upstreamEmissions.Range("G27").Value()
```

```
412         Me.Parameters.urbExtrFedGasPM = upstreamEmissions.Range("G28").Value()
413         Me.Parameters.urbExtrCaliGasPM = upstreamEmissions.Range("G29").Value()
414         Me.Parameters.urbExtrConvDiesPM = upstreamEmissions.Range("G30").Value()
415         Me.Parameters.urbExtrLowDiesPM = upstreamEmissions.Range("G31").Value()
416         Me.Parameters.urbExtrConvGasCO2 = upstreamEmissions.Range("G32").Value()
417         Me.Parameters.urbExtrFedGasCO2 = upstreamEmissions.Range("G33").Value()
418         Me.Parameters.urbExtrCaliGasCO2 = upstreamEmissions.Range("G34").Value()
419         Me.Parameters.urbExtrConvDiesCO2 = upstreamEmissions.Range("G35").Value()
420         Me.Parameters.urbExtrLowDiesCO2 = upstreamEmissions.Range("G36").Value()
421         Me.Parameters.urbExtrConvGasGHG = upstreamEmissions.Range("G37").Value()
422         Me.Parameters.urbExtrFedGasGHG = upstreamEmissions.Range("G38").Value()
423         Me.Parameters.urbExtrCaliGasGHG = upstreamEmissions.Range("G39").Value()
424         Me.Parameters.urbExtrConvDiesGHG = upstreamEmissions.Range("G40").Value()
425         Me.Parameters.urbExtrLowDiesGHG = upstreamEmissions.Range("G41").Value()
426         Me.Parameters.urbTransConvGasCO = upstreamEmissions.Range("H7").Value()
427         Me.Parameters.urbTransFedGasCO = upstreamEmissions.Range("H8").Value()
428         Me.Parameters.urbTransCaliGasCO = upstreamEmissions.Range("H9").Value()
429         Me.Parameters.urbTransConvDiesCO = upstreamEmissions.Range("H10").Value()
430         Me.Parameters.urbTransLowDiesCO = upstreamEmissions.Range("H11").Value()
431         Me.Parameters.urbTransConvGasVOC = upstreamEmissions.Range("H12").Value()
432         Me.Parameters.urbTransFedGasVOC = upstreamEmissions.Range("H13").Value()
433         Me.Parameters.urbTransCaliGasVOC = upstreamEmissions.Range("H14").Value()
434         Me.Parameters.urbTransConvDiesVOC = upstreamEmissions.Range("H15").Value()
435         Me.Parameters.urbTransLowDiesVOC = upstreamEmissions.Range("H16").Value()
436         Me.Parameters.urbTransConvGasNOX = upstreamEmissions.Range("H17").Value()
437         Me.Parameters.urbTransFedGasNOX = upstreamEmissions.Range("H18").Value()
438         Me.Parameters.urbTransCaliGasNOX = upstreamEmissions.Range("H19").Value()
439         Me.Parameters.urbTransConvDiesNOX = upstreamEmissions.Range("H20").Value()
440         Me.Parameters.urbTransLowDiesNOX = upstreamEmissions.Range("H21").Value()
441         Me.Parameters.urbTransConvGasSOX = upstreamEmissions.Range("H22").Value()
442         Me.Parameters.urbTransFedGasSOX = upstreamEmissions.Range("H23").Value()
443         Me.Parameters.urbTransCaliGasSOX = upstreamEmissions.Range("H24").Value()
444         Me.Parameters.urbTransConvDiesSOX = upstreamEmissions.Range("H25").Value()
445         Me.Parameters.urbTransLowDiesSOX = upstreamEmissions.Range("H26").Value()
446         Me.Parameters.urbTransConvGasPM = upstreamEmissions.Range("H27").Value()
447         Me.Parameters.urbTransFedGasPM = upstreamEmissions.Range("H28").Value()
448         Me.Parameters.urbTransCaliGasPM = upstreamEmissions.Range("H29").Value()
449         Me.Parameters.urbTransConvDiesPM = upstreamEmissions.Range("H30").Value()
450         Me.Parameters.urbTransLowDiesPM = upstreamEmissions.Range("H31").Value()
451         Me.Parameters.urbTransConvGasCO2 = upstreamEmissions.Range("H32").Value()
452         Me.Parameters.urbTransFedGasCO2 = upstreamEmissions.Range("H33").Value()
453         Me.Parameters.urbTransCaliGasCO2 = upstreamEmissions.Range("H34").Value()
454         Me.Parameters.urbTransConvDiesCO2 = upstreamEmissions.Range("H35").Value()
455         Me.Parameters.urbTransLowDiesCO2 = upstreamEmissions.Range("H36").Value()
456         Me.Parameters.urbTransConvGasGHG = upstreamEmissions.Range("H37").Value()
457         Me.Parameters.urbTransFedGasGHG = upstreamEmissions.Range("H38").Value()
458         Me.Parameters.urbTransCaliGasGHG = upstreamEmissions.Range("H39").Value()
459         Me.Parameters.urbTransConvDiesGHG = upstreamEmissions.Range("H40").Value()
460         Me.Parameters.urbTransLowDiesGHG = upstreamEmissions.Range("H41").Value()
461         Me.Parameters.urbRefConvGasCO = upstreamEmissions.Range("I7").Value()
462         Me.Parameters.urbRefFedGasCO = upstreamEmissions.Range("I8").Value()
463         Me.Parameters.urbRefCaliGasCO = upstreamEmissions.Range("I9").Value()
464         Me.Parameters.urbRefConvDiesCO = upstreamEmissions.Range("I10").Value()
465         Me.Parameters.urbRefLowDiesCO = upstreamEmissions.Range("I11").Value()
466         Me.Parameters.urbRefConvGasVOC = upstreamEmissions.Range("I12").Value()
467         Me.Parameters.urbRefFedGasVOC = upstreamEmissions.Range("I13").Value()
468         Me.Parameters.urbRefCaliGasVOC = upstreamEmissions.Range("I14").Value()
469         Me.Parameters.urbRefConvDiesVOC = upstreamEmissions.Range("I15").Value()
470         Me.Parameters.urbRefLowDiesVOC = upstreamEmissions.Range("I16").Value()
471         Me.Parameters.urbRefConvGasNOX = upstreamEmissions.Range("I17").Value()
472         Me.Parameters.urbRefFedGasNOX = upstreamEmissions.Range("I18").Value()
473         Me.Parameters.urbRefCaliGasNOX = upstreamEmissions.Range("I19").Value()
474         Me.Parameters.urbRefConvDiesNOX = upstreamEmissions.Range("I20").Value()
475         Me.Parameters.urbRefLowDiesNOX = upstreamEmissions.Range("I21").Value()
476         Me.Parameters.urbRefConvGasSOX = upstreamEmissions.Range("I22").Value()
477         Me.Parameters.urbRefFedGasSOX = upstreamEmissions.Range("I23").Value()
478         Me.Parameters.urbRefCaliGasSOX = upstreamEmissions.Range("I24").Value()
479         Me.Parameters.urbRefConvDiesSOX = upstreamEmissions.Range("I25").Value()
480         Me.Parameters.urbRefLowDiesSOX = upstreamEmissions.Range("I26").Value()
```

```
481         Me.Parameters.urbRefConvGasPM = upstreamEmissions.Range("I27").Value()
482         Me.Parameters.urbRefFedGasPM = upstreamEmissions.Range("I28").Value()
483         Me.Parameters.urbRefCaliGasPM = upstreamEmissions.Range("I29").Value()
484         Me.Parameters.urbRefConvDiesPM = upstreamEmissions.Range("I30").Value()
485         Me.Parameters.urbRefLowDiesPM = upstreamEmissions.Range("I31").Value()
486         Me.Parameters.urbRefConvGasCO2 = upstreamEmissions.Range("I32").Value()
487         Me.Parameters.urbRefFedGasCO2 = upstreamEmissions.Range("I33").Value()
488         Me.Parameters.urbRefCaliGasCO2 = upstreamEmissions.Range("I34").Value()
489         Me.Parameters.urbRefConvDiesCO2 = upstreamEmissions.Range("I35").Value()
490         Me.Parameters.urbRefLowDiesCO2 = upstreamEmissions.Range("I36").Value()
491         Me.Parameters.urbRefConvGasGHG = upstreamEmissions.Range("I37").Value()
492         Me.Parameters.urbRefFedGasGHG = upstreamEmissions.Range("I38").Value()
493         Me.Parameters.urbRefCaliGasGHG = upstreamEmissions.Range("I39").Value()
494         Me.Parameters.urbRefConvDiesGHG = upstreamEmissions.Range("I40").Value()
495         Me.Parameters.urbRefLowDiesGHG = upstreamEmissions.Range("I41").Value()
496         Me.Parameters.urbTSDConvGasCO = upstreamEmissions.Range("J7").Value()
497         Me.Parameters.urbTSDFedGasCO = upstreamEmissions.Range("J8").Value()
498         Me.Parameters.urbTSDCaliGasCO = upstreamEmissions.Range("J9").Value()
499         Me.Parameters.urbTSDConvDiesCO = upstreamEmissions.Range("J10").Value()
500         Me.Parameters.urbTSDLowDiesCO = upstreamEmissions.Range("J11").Value()
501         Me.Parameters.urbTSDConvGasVOC = upstreamEmissions.Range("J12").Value()
502         Me.Parameters.urbTSDFedGasVOC = upstreamEmissions.Range("J13").Value()
503         Me.Parameters.urbTSDCaliGasVOC = upstreamEmissions.Range("J14").Value()
504         Me.Parameters.urbTSDConvDiesVOC = upstreamEmissions.Range("J15").Value()
505         Me.Parameters.urbTSDLowDiesVOC = upstreamEmissions.Range("J16").Value()
506         Me.Parameters.urbTSDConvGasNOX = upstreamEmissions.Range("J17").Value()
507         Me.Parameters.urbTSDFedGasNOX = upstreamEmissions.Range("J18").Value()
508         Me.Parameters.urbTSDCaliGasNOX = upstreamEmissions.Range("J19").Value()
509         Me.Parameters.urbTSDConvDiesNOX = upstreamEmissions.Range("J20").Value()
510         Me.Parameters.urbTSDLowDiesNOX = upstreamEmissions.Range("J21").Value()
511         Me.Parameters.urbTSDConvGasSOX = upstreamEmissions.Range("J22").Value()
512         Me.Parameters.urbTSDFedGasSOX = upstreamEmissions.Range("J23").Value()
513         Me.Parameters.urbTSDCaliGasSOX = upstreamEmissions.Range("J24").Value()
514         Me.Parameters.urbTSDConvDiesSOX = upstreamEmissions.Range("J25").Value()
515         Me.Parameters.urbTSDLowDiesSOX = upstreamEmissions.Range("J26").Value()
516         Me.Parameters.urbTSDConvGasPM = upstreamEmissions.Range("J27").Value()
517         Me.Parameters.urbTSDFedGasPM = upstreamEmissions.Range("J28").Value()
518         Me.Parameters.urbTSDCaliGasPM = upstreamEmissions.Range("J29").Value()
519         Me.Parameters.urbTSDConvDiesPM = upstreamEmissions.Range("J30").Value()
520         Me.Parameters.urbTSDLowDiesPM = upstreamEmissions.Range("J31").Value()
521         Me.Parameters.urbTSDConvGasCO2 = upstreamEmissions.Range("J32").Value()
522         Me.Parameters.urbTSDFedGasCO2 = upstreamEmissions.Range("J33").Value()
523         Me.Parameters.urbTSDCaliGasCO2 = upstreamEmissions.Range("J34").Value()
524         Me.Parameters.urbTSDConvDiesCO2 = upstreamEmissions.Range("J35").Value()
525         Me.Parameters.urbTSDLowDiesCO2 = upstreamEmissions.Range("J36").Value()
526         Me.Parameters.urbTSDConvGasGHG = upstreamEmissions.Range("J37").Value()
527         Me.Parameters.urbTSDFedGasGHG = upstreamEmissions.Range("J38").Value()
528         Me.Parameters.urbTSDCaliGasGHG = upstreamEmissions.Range("J39").Value()
529         Me.Parameters.urbTSDConvDiesGHG = upstreamEmissions.Range("J40").Value()
530         Me.Parameters.urbTSDLowDiesGHG = upstreamEmissions.Range("J41").Value()
531
532         ' assign values to vars in fleetParameters worksheet
533         Me.Parameters.percentSalesPreviousYear = fleetParameters.Range("B5").Value()
534         Me.Parameters.percentSalesCurrentYear = fleetParameters.Range("B6").Value()
535         Me.Parameters.percentSalesFollowingYear = fleetParameters.Range("B7").Value()
536         Me.Parameters.percentLDGT1 = fleetParameters.Range("B9").Value()
537         Me.Parameters.percentLDGT2 = fleetParameters.Range("B10").Value()
538         Me.Parameters.percentLDDT12 = fleetParameters.Range("B11").Value()
539         Me.Parameters.percentLDGT3 = fleetParameters.Range("B12").Value()
540         Me.Parameters.percentLDGT4 = fleetParameters.Range("B13").Value()
541         Me.Parameters.percentLDDT34 = fleetParameters.Range("B14").Value()
542         Me.Parameters.gapTestOnRoad = fleetParameters.Range("B15").Value()
543         Me.Parameters.fuelTankLT6000 = fleetParameters.Range("B17").Value()
544         Me.Parameters.fuelTankLT60008500 = fleetParameters.Range("B18").Value()
545         Me.Parameters.fuelTankLT8500 = fleetParameters.Range("B19").Value()
546         Me.Parameters.fuelTankCars = fleetParameters.Range("B20").Value()
547
548         ' assign values to vars in economicValues worksheet
549         Me.Parameters.reboundEffect = economicValues.Range("B4").Value()
```

```
550         Me.Parameters.discountRate = economicValues.Range("B5").Value()
551         Me.Parameters.monopsonyCost = economicValues.Range("B7").Value()
552         Me.Parameters.priceShockCost = economicValues.Range("B8").Value()
553         Me.Parameters.militarySecurityCost = economicValues.Range("B9").Value()
554         Me.Parameters.totalEconomicCost = Me.Parameters.monopsonyCost + _
555                                           Me.Parameters.priceShockCost + Me.Parameters.
    militarySecurityCost
556         Me.Parameters.totalEconomicCostBBL = Me.Parameters.totalEconomicCost * 42
557         Me.Parameters.congestionCost = economicValues.Range("B13").Value()
558         Me.Parameters.accidentCost = economicValues.Range("B14").Value()
559         Me.Parameters.noiseCost = economicValues.Range("B15").Value()
560         Me.Parameters.totalExternalCost = Me.Parameters.congestionCost + _
561                                           Me.Parameters.accidentCost + Me.Parameters.noiseCost
562         Me.Parameters.damageCostCO = economicValues.Range("B18").Value()
563         Me.Parameters.damageCostVOC = economicValues.Range("B19").Value()
564         Me.Parameters.damageCostNOX = economicValues.Range("B20").Value()
565         Me.Parameters.damageCostPM = economicValues.Range("B21").Value()
566         Me.Parameters.damageCostSO2 = economicValues.Range("B22").Value()
567         Me.Parameters.damageCostC = economicValues.Range("B23").Value()
568         Me.Parameters.damageCostCO2 = economicValues.Range("B24").Value()
569         Me.Parameters.travelValuePerson = economicValues.Range("B26").Value()
570         Me.Parameters.averageVehicleOccupancy = economicValues.Range("B27").Value()
571         Me.Parameters.travelValueVehicle = economicValues.Range("B28").Value()
572
573         'assign values to arrays in forecastData worksheet
574         'year 2000 will be found in array(0,0)
575         f = 0
576         For g = 0 To 25
577             Me.Parameters.gasSalesAuto(f, g) = forecastData.Cells(6, g + 3).Value()
578             Me.Parameters.diesSalesAuto(f, g) = forecastData.Cells(7, g + 3).Value()
579             Me.Parameters.ethSalesAuto(f, g) = forecastData.Cells(8, g + 3).Value()
580             Me.Parameters.totSalesAuto(f, g) = forecastData.Cells(9, g + 3).Value()
581             Me.Parameters.gasSalesLT(f, g) = forecastData.Cells(11, g + 3).Value()
582             Me.Parameters.diesSalesLT(f, g) = forecastData.Cells(12, g + 3).Value()
583             Me.Parameters.ethSalesLT(f, g) = forecastData.Cells(13, g + 3).Value()
584             Me.Parameters.totSalesLT(f, g) = forecastData.Cells(14, g + 3).Value()
585             Me.Parameters.ratingEPAAuto(f, g) = forecastData.Cells(16, g + 3).Value()
586             Me.Parameters.ratingEPALT(f, g) = forecastData.Cells(17, g + 3).Value()
587             Me.Parameters.ratioMPGAuto(f, g) = forecastData.Cells(19, g + 3).Value()
588             Me.Parameters.ratioMPGLT(f, g) = forecastData.Cells(20, g + 3).Value()
589         Next
590
591         For g = 0 To 25
592             Me.Parameters.fuelPriceGas(g) = forecastData.Cells(22, g + 3).Value()
593             Me.Parameters.fuelPriceDiesel(g) = forecastData.Cells(23, g + 3).Value()
594             Me.Parameters.fuelTaxGasFed(g) = forecastData.Cells(26, g + 3).Value()
595             Me.Parameters.fuelTaxGasState(g) = forecastData.Cells(27, g + 3).Value()
596             Me.Parameters.fuelTaxGasTot(g) = forecastData.Cells(28, g + 3).Value()
597             Me.Parameters.fuelTaxDiesFed(g) = forecastData.Cells(30, g + 3).Value()
598             Me.Parameters.fuelTaxDiesState(g) = forecastData.Cells(31, g + 3).Value()
599             Me.Parameters.fuelTaxDiesTot(g) = forecastData.Cells(32, g + 3).Value()
600             Me.Parameters.fuelPretaxGas(g) = forecastData.Cells(34, g + 3).Value()
601             Me.Parameters.fuelPretaxDies(g) = forecastData.Cells(35, g + 3).Value()
602         Next
603
604
605
606
607
608
609
610
611
612
613
614
615                                                                                    )
616
617
```

```
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661         ' get min and max model and calendar rates years--any rates sheet will do
662         Me.GetMinAndMaxRatesYears(Me.coRatesGas)
663
664         ReadRatesSheets(coRatesGas, Me.Parameters.coGasLDGV, _
665                         Me.Parameters.coGasLDGT1, _
666                         Me.Parameters.coGasLDGT2, _
667                         Me.Parameters.coGasHDGV2b)
668
669         ReadRatesSheets(coRatesGasRFG, Me.Parameters.coGasRFGLDGV, _
670                         Me.Parameters.coGasRFGLDGT1, _
671                         Me.Parameters.coGasRFGLDGT2, _
672                         Me.Parameters.coGasRFGHDGV2b)
673
674         ReadRatesSheets(coRatesDies, Me.Parameters.coDiesLDDV, _
675                         Me.Parameters.coDiesLDDT1, _
676                         Me.Parameters.coDiesLDDT2, _
677                         Me.Parameters.coDiesHDDV2b)
678
679         ReadRatesSheets(vocRatesGas, Me.Parameters.vocGasLDGV, _
680                         Me.Parameters.vocGasLDGT1, _
681                         Me.Parameters.vocGasLDGT2, _
682                         Me.Parameters.vocGasHDGV2b)
683
684         ReadRatesSheets(vocRatesGasRFG, Me.Parameters.vocGasRFGLDGV, _
685                         Me.Parameters.vocGasRFGLDGT1, _
686                         Me.Parameters.vocGasRFGLDGT2, _
```

```
687                         Me.Parameters.vocGasRFGHDGV2b)
688
689         ReadRatesSheets(vocRatesDies, Me.Parameters.vocDiesLDDV, _
690                         Me.Parameters.vocDiesLDDT1, _
691                         Me.Parameters.vocDiesLDDT2, _
692                         Me.Parameters.vocDiesHDDV2b)
693
694         ReadRatesSheets(noxRatesGas, Me.Parameters.noxGasLDGV, _
695                         Me.Parameters.noxGasLDGT1, _
696                         Me.Parameters.noxGasLDGT2, _
697                         Me.Parameters.noxGasHDGV2b)
698
699         ReadRatesSheets(noxRatesGasRFG, Me.Parameters.noxGasRFGLDGV, _
700                         Me.Parameters.noxGasRFGLDGT1, _
701                         Me.Parameters.noxGasRFGLDGT2, _
702                         Me.Parameters.noxGasRFGHDGV2b)
703
704         ReadRatesSheets(noxRatesDies, Me.Parameters.noxDiesLDDV, _
705                         Me.Parameters.noxDiesLDDT1, _
706                         Me.Parameters.noxDiesLDDT2, _
707                         Me.Parameters.noxDiesHDDV2b)
708
709         ReadRatesSheets(pmRatesGas, Me.Parameters.pmGasLDGV, _
710                         Me.Parameters.pmGasLDGT1, _
711                         Me.Parameters.pmGasLDGT2, _
712                         Me.Parameters.pmGasHDGV2b)
713
714         ReadRatesSheets(pmRatesGasRFG, Me.Parameters.pmGasRFGLDGV, _
715                         Me.Parameters.pmGasRFGLDGT1, _
716                         Me.Parameters.pmGasRFGLDGT2, _
717                         Me.Parameters.pmGasRFGHDGV2b)
718
719         ReadRatesSheets(pmRatesDies, Me.Parameters.pmDiesLDDV, _
720                         Me.Parameters.pmDiesLDDT1, _
721                         Me.Parameters.pmDiesLDDT2, _
722                         Me.Parameters.pmDiesHDDV2b)
723
724         ReadRatesSheets(so2RatesGas, Me.Parameters.so2GasLDGV, _
725                         Me.Parameters.so2GasLDGT1, _
726                         Me.Parameters.so2GasLDGT2, _
727                         Me.Parameters.so2GasHDGV2b)
728
729         ReadRatesSheets(so2RatesGasRFG, Me.Parameters.so2GasRFGLDGV, _
730                         Me.Parameters.so2GasRFGLDGT1, _
731                         Me.Parameters.so2GasRFGLDGT2, _
732                         Me.Parameters.so2GasRFGHDGV2b)
733
734         ReadRatesSheets(so2RatesDies, Me.Parameters.so2DiesLDDV, _
735                         Me.Parameters.so2DiesLDDT1, _
736                         Me.Parameters.so2DiesLDDT2, _
737                         Me.Parameters.so2DiesHDDV2b)
738
739         ' get kmy values
740         Me.Parameters.Kmy = kMyTypeClass.GetKmyValues(Me.Parameters)
741
742         ' copy the parameters data into the global cafe.parameters variable
743         ParametersClass.Copy(CAFE.Parameters, Me.Parameters)
744
745         ' close the file
746         Me.CloseFile()
747         MyBase.ValidFile = True
748     End Sub
749
750     '
751     ' other methods
752     '
753
754     '''
755     ' Closes the fles
```

```
756     '
757     Private Function CloseFile()
758         ' close the file
759         ParametersFile.CloseFile()
760         ParametersFile = Nothing
761     End Function
762
763     '''
764     ' This function loads all required worksheets from the input parameters file.
765     '
766     Private Function LoadWorksheets() As Boolean
767         Dim ActSheet As Excel.Worksheet
768         Dim SheetName As String
769         Dim nSheets As Integer
770         Dim s As Integer
771         Dim FoundComplianceSheets(4) As Boolean      ' 4 sheets:  1-4
772         Dim FoundEffectsSheets(22) As Boolean        ' 22 sheets: 1-22
773         Dim FoundAllSheets As Boolean                ' checks if all sheets are present
774
775         nSheets = Wb.Worksheets.Count
776         FoundAllSheets = True
777
778         For s = 1 To nSheets
779             ActSheet = ParametersFile.SetSheet(s)
780             SheetName = ActSheet.Name
781
782             If Not IsNothing(SheetName) Then
783                 '
784                 ' load compliance model data worksheets
785                 '
786                 LoadWorksheet(SheetName, "General", s, Me.GeneralParameters,
    FoundComplianceSheets(1))
787                 LoadWorksheet(SheetName, "Cost Allocation", s, Me.CostAllocation,
    FoundComplianceSheets(2))
788                 LoadWorksheet(SheetName, "MY02LTWeight", s, Me.MY02LTWeight,
    FoundComplianceSheets(3))
789                 LoadWorksheet(SheetName, "Willing To Pay Fines", s, Me.WillingToPayFines,
    FoundComplianceSheets(4))
790                 LoadWorksheet(SheetName, "NMNL", s, Me.NMNL, Me.NMNLIncluded)
791                 CAFE.NMNLIncluded = Me.NMNLIncluded
792
793                 '
794                 ' load effects model data worksheets
795                 '
796                 LoadWorksheet(SheetName, "Vehicle Age Data", s, Me.vehicleAgeData,
    FoundEffectsSheets(1))
797                 LoadWorksheet(SheetName, "Mileage Equations", s, Me.mileageEquations,
    FoundEffectsSheets(2))
798                 LoadWorksheet(SheetName, "Fuel Properties", s, Me.fuelProperties,
    FoundEffectsSheets(3))
799                 LoadWorksheet(SheetName, "Upstream Emissions", s, Me.upstreamEmissions,
    FoundEffectsSheets(4))
800                 LoadWorksheet(SheetName, "Fleet Parameters", s, Me.fleetParameters,
    FoundEffectsSheets(5))
801                 LoadWorksheet(SheetName, "Economic Values", s, Me.economicValues,
    FoundEffectsSheets(6))
802                 LoadWorksheet(SheetName, "Forecast Data", s, Me.forecastData,
    FoundEffectsSheets(7))
803                 LoadWorksheet(SheetName, "CO Rates - Gas", s, Me.coRatesGas,
    FoundEffectsSheets(8))
804                 LoadWorksheet(SheetName, "CO Rates - Gas RFG", s, Me.coRatesGasRFG,
    FoundEffectsSheets(9))
805                 LoadWorksheet(SheetName, "CO Rates - Diesel", s, Me.coRatesDies,
    FoundEffectsSheets(10))
806                 LoadWorksheet(SheetName, "VOC Rates - Gas", s, Me.vocRatesGas,
    FoundEffectsSheets(11))
807                 LoadWorksheet(SheetName, "VOC Rates - Gas RFG", s, Me.vocRatesGasRFG,
    FoundEffectsSheets(12))
808                 LoadWorksheet(SheetName, "VOC Rates - Diesel", s, Me.vocRatesDies,
```

```
    FoundEffectsSheets(13))
809             LoadWorksheet(SheetName, "NOx Rates - Gas", s, Me.noxRatesGas,
    FoundEffectsSheets(14))
810             LoadWorksheet(SheetName, "NOx Rates - Gas RFG", s, Me.noxRatesGasRFG,
    FoundEffectsSheets(15))
811             LoadWorksheet(SheetName, "NOx Rates - Diesel", s, Me.noxRatesDies,
    FoundEffectsSheets(16))
812             LoadWorksheet(SheetName, "PM Rates - Gas", s, Me.pmRatesGas,
    FoundEffectsSheets(17))
813             LoadWorksheet(SheetName, "PM Rates - Gas RFG", s, Me.pmRatesGasRFG,
    FoundEffectsSheets(18))
814             LoadWorksheet(SheetName, "PM Rates - Diesel", s, Me.pmRatesDies,
    FoundEffectsSheets(19))
815             LoadWorksheet(SheetName, "SO2 Rates - Gas", s, Me.so2RatesGas,
    FoundEffectsSheets(20))
816             LoadWorksheet(SheetName, "SO2 Rates - Gas RFG", s, Me.so2RatesGasRFG,
    FoundEffectsSheets(21))
817             LoadWorksheet(SheetName, "SO2 Rates - Diesel", s, Me.so2RatesDies,
    FoundEffectsSheets(22))
818
819             LoadWorksheet(SheetName, "Safety Values", s, Me.safetyValues, Me.
    SafetyIsHere)
820             CAFE.SafetyEffectsLoaded = SafetyIsHere
821         End If
822     Next s
823
824     ' check if all required compliance model worksheets have been loaded
825     For s = 1 To 4
826         ' if any compliance sheet is not present, set found all to false
827         If Not FoundComplianceSheets(s) Then
828             MsgBox("Not all of the required compliance model worksheets were found.", _
829                    MsgBoxStyle.Information, "Missing Compliance Worksheets.")
830             FoundAllSheets = False
831             Exit For
832         End If
833     Next s
834
835     ' check if all required effects worksheets have been loaded
836     For s = 1 To 22
837         ' if any effects sheet is not present, set found all to false
838         If Not FoundEffectsSheets(s) Then
839             MsgBox("Not all of the required effects model worksheets were found.", _
840                    MsgBoxStyle.Information, "Missing Effects Worksheets.")
841             FoundAllSheets = False
842             Exit For
843         End If
844     Next s
845
846     Return FoundAllSheets
847 End Function
848
849 '''
850 ' This method loads a single worksheet based on the parameters specified.
851 '
852 ' Params:
853 '   SheetName   - the name of the current active worksheet
854 '   DesiredName - the name of the comparing worksheet
855 '   SheetIndex  - the index of the active worksheet
856 '   SheetObject - the object to which to assign the active sheet if the SheetName matches
     the
857 '                 DesiredName
858 '   FoundSheet  - specifies whether the given SheetObject has already been found
859 '
860 Private Sub LoadWorksheet(ByVal SheetName As String, _
861                           ByVal DesiredName As String, _
862                           ByVal SheetIndex As Integer, _
863                           ByRef SheetObject As Object, _
864                           ByRef FoundSheet As Boolean)
865     '
```

```
866         If UCase(SheetName) = UCase(DesiredName) Then
867             SheetObject = Me.ParametersFile.GetSheet(SheetIndex)
868             FoundSheet = True
869         End If
870     End Sub
871
872     '''
873     ' This method loads the raw data from the current active worksheet within the given bounds if
874     ' the desired worksheet name matches the active worksheet name.
875     '
876     Private Sub LoadRawData(ByVal SheetObject As Excel.Worksheet, _
877                             ByRef RawData(,) As Object, _
878                             ByVal x0 As Integer, ByVal y0 As Integer, _
879                             ByVal x1 As Integer, ByVal y1 As Integer)
880         '
881         RawData = Me.GetRawData(SheetObject, SheetObject.Cells(x0, y0), SheetObject.Cells(x1, y1))
882     End Sub
883
884     '''
885     ' Use the willing to pay fines worksheet to get the number of available manufacturers and
886     ' number of available model years.
887     '
888     Private Function GetNumMfgAndMY() As Integer
889         Dim i As Integer
890         Dim RawData(,) As Object = GetRawData(Me.WillingToPayFines, 3)
891         Me.nMfg = RawData(0, 0) - 1
892         Me.nMY = RawData(0, 1) - 1
893
894         Me.Parameters.nMfg = Me.nMfg
895         Me.Parameters.wtpfnMY = Me.nMY
896
897         ' find model years
898         ReDim Me.Parameters.wtpfMY(Me.nMY - 1)
899         For i = 0 To Me.nMY - 1
900             Me.Parameters.wtpfMY(i) = GetDouble(RawData, 1, 2 + i)
901         Next i
902
903         ' find min and max years
904         Me.Parameters.wtpfMinYear = Me.Parameters.wtpfMY(0)
905         Me.Parameters.wtpfMaxYear = Me.Parameters.wtpfMY(Me.nMY - 1)
906     End Function
907
908     '''
909     '
910     '
911     Private Sub GetMinAndMaxRatesYears(ByVal RateSheet As Object)
912         Dim i, j As Integer
913         Dim Wdata_MY(,), Wdata_CY(,) As Object
914         Dim SR_MY, SR_CY As Excel.Range
915         Dim ER_MY, ER_CY As Excel.Range
916
917         SR_MY = RateSheet.Range("A4")
918         ER_MY = RateSheet.Range("A29")
919
920         SR_CY = RateSheet.Range("B3")
921         ER_CY = RateSheet.Range("AA3")
922
923         Wdata_MY = CType(RateSheet, Excel.Worksheet).Range(SR_MY, ER_MY).Resize.Value
924         Wdata_CY = CType(RateSheet, Excel.Worksheet).Range(SR_CY, ER_CY).Resize.Value
925
926         ' load available my and cy
927         ReDim Me.RatesMY(25)
928         ReDim Me.RatesCY(25)
929
930         For i = 0 To 25
931             Me.RatesMY(i) = Me.GetInteger(Wdata_MY(i + 1, 1))
932             Me.RatesCY(i) = Me.GetInteger(Wdata_CY(1, i + 1))
```

```
933         Next i
934
935         ' set min and max my and cy
936         Me.Parameters.RatesMinMY = Me.RatesMY(0)
937         Me.Parameters.RatesMaxMY = Me.RatesMY(25)
938         Me.Parameters.RatesMinCY = Me.RatesCY(0)
939         Me.Parameters.RatesMaxCY = Me.RatesCY(25)
940     End Sub
941
942     Private Sub ReadRatesSheets(ByVal RateSheet As Object, _
943                                 ByRef A_table(,) As Double, _
944                                 ByRef B_table(,) As Double, _
945                                 ByRef C_table(,) As Double, _
946                                 ByRef D_table(,) As Double)
947         '
948         ' Assume that the tables are fixed length -- from A1 to AA119 in the worksheet
949         Dim Wdata(,) As Object
950         Dim SR As Excel.Range
951         Dim ER As Excel.Range
952         Dim MYIdx As Integer = CAFE.GMYI(Me.Parameters.RatesMinMY)
953         Dim CYIdx As Integer = CAFE.GMYI(Me.Parameters.RatesMinCY)
954         Dim MaxMYIdx As Integer = CAFE.GMYI(Me.Parameters.RatesMaxMY)
955         Dim MaxCYIdx As Integer = CAFE.GMYI(Me.Parameters.RatesMaxCY)
956
957         ' Establish "start" and "end" points for reading the worksheet
958         ER = RateSheet.Range("AA119")
959         SR = RateSheet.Range("A1")
960
961         ' read the worksheet into an object array
962         Wdata = RateSheet.Range(SR, ER).Resize.Value
963
964         ' resize the arrays
965         ReDim A_table(MaxMYIdx, MaxCYIdx)
966         ReDim B_table(MaxMYIdx, MaxCYIdx)
967         ReDim C_table(MaxMYIdx, MaxCYIdx)
968         ReDim D_table(MaxMYIdx, MaxCYIdx)
969
970         ' Digest the array
971         For f As Integer = 0 To 25
972             For g As Integer = 0 To 25
973                 A_table(MYIdx + f, CYIdx + g) = GetDouble(Wdata(f + 4, g + 2))
974                 If IsNothing(B_table) = False Then
975                     B_table(MYIdx + f, CYIdx + g) = GetDouble(Wdata(f + 34, g + 2))
976                 End If
977                 If IsNothing(C_table) = False Then
978                     C_table(MYIdx + f, CYIdx + g) = GetDouble(Wdata(f + 64, g + 2))
979                 End If
980                 If IsNothing(D_table) = False Then
981                     D_table(MYIdx + f, CYIdx + g) = GetDouble(Wdata(f + 94, g + 2))
982                 End If
983             Next g
984         Next f
985     End Sub
986
987     Private RatesMY() As Integer
988     Private RatesCY() As Integer
989
990 End Class
991
```

```
Public Class RawScenariosClass
    Inherits RawInput


    Public Scenarios As ScenariosClass

    Private x0, y0, x1, y1 As Integer
    ' the min and max years for each scenario
    Private MinYear, MaxYear As Double

    ' the scenarios file
    Private ScenariosFile As C_Cafe_InFile
    ' the scenarios file name
    Private fName As String

#Region " Data "
    ' the scenario page
    Private ScenarioIdx As Integer = 0

    Private nMY As Integer
    Private nLT_RegClass As Integer
    Private nCoefficient As Integer

    Private RawScenarioDescription(,) As Object
    Private RawAuto_CAFE_STD(,) As Object
    Private RawApplicabilityOfLTProgram(,) As Object
    Private RawLT_RegClassBoundaries(,) As Object
    Private RawLT_RegClassFlatSTD(,) As Object
    Private RawLT_Trade(,) As Object
    Private RawLT_FunctionType(,) As Object
    Private RawLT_FunctionCoefficient(,) As Object
    Private RawHLT_FlatSTD(,) As Object
    Private RawTransitionalFlatSTD(,) As Object
    Private RawCAFE_FineRate(,) As Object

    ' determines whether or not at least one scenarios sheet has been found
    Private FoundScenarioSheet As Boolean = False
#End Region

    Public Sub New()
    End Sub

    Public Sub New(ByVal fName As String)
        ' get parameters file
        Me.ScenariosFile = New C_Cafe_InFile
        Me.fName = fName

        Dim ActSheet As Excel.Worksheet
        Dim nSheets As Integer
        Dim s As Integer

        Dim SheetName As String

        ' stores the error msg to display to user in case of error
        Dim MsgString As String = ""

        If Not Me.ScenariosFile.OpenFile(fName) Then
            ' could not open file for any reason, prompt user and exit...
            MsgBox("An error has occured while opening the scenarios input file.  " & _
                   "Please try again.", MsgBoxStyle.Exclamation, "Error Opening File")
            Exit Sub
        End If

        Me.Wb = Me.ScenariosFile.GetWorkbookObject
        nSheets = Wb.Worksheets.Count

        ' parse wb title
        If Not Me.ParseWbTitle("CAFE Scenarios file", 1.0) Then
            ' if wb title or version did not match desired once
            ' close file and exit sub
```

```
71              Me.ScenariosFile.CloseFile()
72              Me.ScenariosFile = Nothing
73              Exit Sub
74          End If
75
76          Me.ScenariosFile.Activate()
77          Scenarios = New ScenariosClass
78
79          ' loop through each sheet and load the necessary data
80          For s = 1 To nSheets
81              ActSheet = Me.ScenariosFile.SetSheet(s)
82              SheetName = ActSheet.Name
83
84              If Not IsNothing(SheetName) Then
85                  If LCase(SheetName).StartsWith("scen") Then
86                      ' Scenario specific Parameters
87                      If Not Me.GetScenarioParameters(ActSheet) Then
88                          MsgBox("Failed to load file because of missing data.", _
89                                 MsgBoxStyle.Exclamation, "Missing data.")
90                          CloseFile()
91                          Exit Sub
92                      End If
93                      Me.FoundScenarioSheet = True
94                  Else
95                      ' ignore all other sheet
96                      '
97                  End If
98              End If
99          Next s
100
101         ' get error prompt
102         MsgString = NewLine & NewLine
103         If Not Me.FoundScenarioSheet Then MsgString += "    - CAFE Scenario Definition
        Worksheet (scen******)" & NewLine
104
105         If Not Me.FoundScenarioSheet Then
106             ' not all of the required sheets were found, prompt user
107             MsgBox("The following required worksheets are missing: " & MsgString & _
108                    NewLine & "Please try loading a different file.", _
109                    MsgBoxStyle.Exclamation, "Missing worksheets")
110             ' close the file and exit sub
111             CloseFile()
112             Exit Sub
113         End If
114
115         ' set nScenarios counter
116         Me.Scenarios.nScenarios = ScenarioIdx
117
118         ' copy the scenarios data into the global cafe.scenarios variable
119         ScenariosClass.Copy(CAFE.Scenarios, Scenarios)
120
121         ' close the file
122         CloseFile()
123         MyBase.ValidFile = True
124     End Sub
125
126     Private Function CloseFile()
127         Me.ScenariosFile.CloseFile()
128         Me.ScenariosFile = Nothing
129     End Function
130
131     Public Function GetScenarioParameters(ByRef ActSheet As Excel.Worksheet) As Boolean
132         Dim MsgString As String = ""
133
134         ' first, get the essential params data, such as # of MY's and RC's
135         GetNumMY(ActSheet)
136         GetNumLT_RC(ActSheet)
137         GetNumCoefficient(ActSheet)
138
139         ' next, find the data for each parameter
```

```
140         FindScenarioDescription(ActSheet)
141         FindAuto_CAFE_STD(ActSheet)
142         FindRawApplicabilityOfLTProgram(ActSheet)
143         FindLT_RegClassBoundaries(ActSheet)
144         FindLT_RegClassFlatSTD(ActSheet)
145         FindLT_Trade(ActSheet)
146         FindLT_FunctionType(ActSheet)
147         FindLT_FunctionCoefficient(ActSheet)
148         FindHLT_FlatSTD(ActSheet)
149         FindTransitionalFlatSTD(ActSheet)
150         FindCAFE_FineRate(ActSheet)
151
152         ' after data is loaded, verify...
153         VerifyRawData(RawScenarioDescription, "Scenario Description", MsgString, 1, 1)
154         VerifyRawData(RawAuto_CAFE_STD, "Passenger Automobile CAFE Standard", MsgString, 1, ↙
        1)
155         VerifyRawData(RawApplicabilityOfLTProgram, "Applicability of Light Truck Program", ↙
    MsgString, 1, 1)
156         VerifyRawData(RawLT_RegClassBoundaries, "LT Reg. Class Boundaries", MsgString, 1, ↙
    1)
157         VerifyRawData(RawLT_RegClassFlatSTD, "Light Truck Flat Standard Value", MsgString, ↙
    1, 1)
158         VerifyRawData(RawLT_Trade, "Trading Between LT Classes", MsgString, 1, 1)
159         VerifyRawData(RawLT_FunctionType, "LT Std. Functional Form", MsgString, 1, 1)
160         VerifyRawData(RawLT_FunctionCoefficient, "LT Functional Form Coefficients", ↙
    MsgString, 1, 1)
161         VerifyRawData(RawHLT_FlatSTD, "HLT Flat Standard", MsgString, 1, 1)
162         VerifyRawData(RawTransitionalFlatSTD, "Transitional Flat Standard", MsgString, 1, ↙
    1)
163         VerifyRawData(RawCAFE_FineRate, "CAFE Fine Rate", MsgString, 1, 1)
164
165         If MsgString <> "" Then
166             ' one or more of the params is not found
167             MsgBox("The following parameter section(s) were not found: " & _
168                    NewLine & NewLine & MsgString & NewLine & "Possibly the input " & _
169                    "parameters file (" & fName & ") is not properly alligned.", _
170                    MsgBoxStyle.Exclamation, "Missing Parameters Information")
171
172             ' exit the function without parsing data
173             Return False
174         End If
175
176         ' after verification passed, parse data and store it into ParametersClass.
177         If IsNothing(Me.Scenarios) Then
178             Me.Scenarios = New ScenariosClass
179         End If
180         ' add and initialize a new scenario in Scenarios
181         ReDim Preserve Me.Scenarios.Scenarios(ScenarioIdx)
182         Me.Scenarios.Scenarios(Me.ScenarioIdx) = New ScenarioClass
183
184         Me.Scenarios.Scenarios(Me.ScenarioIdx).nMY = Me.nMY
185         Me.Scenarios.Scenarios(Me.ScenarioIdx).nCoefficients = Me.nCoefficient
186
187         ParseScenarioDescription(ActSheet)
188         ParseAuto_CAFE_STD(ActSheet)
189         ParseRawApplicabilityOfLTProgram(ActSheet)
190         ParseLT_RegClassBoundaries(ActSheet)
191         ParseLT_RegClassFlatSTD(ActSheet)
192         ParseLT_Trade(ActSheet)
193         ParseLT_FunctionType(ActSheet)
194         ParseLT_FunctionCoefficient(ActSheet)
195         ParseHLT_FlatSTD(ActSheet)
196         ParseTransitionalFlatSTD(ActSheet)
197         ParseCAFE_FineRate(ActSheet)
198
199         Me.ScenarioIdx += 1     ' increment scenario index
200
201         ' all data was loaded successfully, return true
202         Return True
203     End Function
```

```
204
205     Private Function GetNumMY(ByRef ActSheet As Excel.Worksheet) As Integer
206         Dim RawData(,) As Object = GetRawRow(ActSheet, 6)
207         nMY = RawData(0, 1) - 2
208     End Function
209
210     Private Function GetNumLT_RC(ByRef ActSheet As Excel.Worksheet) As Integer
211         Dim i As Integer = 0
212         Dim RawData(,) As Object
213
214         Do
215             RawData = GetRawRow(ActSheet, 18 + i)
216             i += 1
217         Loop Until IsNothing(RawData)
218
219         nLT_RegClass = i - 1
220     End Function
221
222     Private Function GetNumCoefficient(ByRef ActSheet As Excel.Worksheet) As Integer
223         Dim i As Integer = 0
224         Dim start As Integer = 33 + (2 * nLT_RegClass)
225         Dim RawData(,) As Object
226
227         Do
228             RawData = GetRawRow(ActSheet, start + i)
229             i += 1
230         Loop Until IsNothing(RawData)
231
232         nCoefficient = i - 1
233     End Function
234
235
236  Find Data methods
237
238
239 #Region " Parse Data methods "
240
241     Public Function ParseScenarioDescription(ByRef ActSheet As Excel.Worksheet) As Boolean
242         Me.Scenarios.Scenarios(ScenarioIdx).ScenarioDescription = Me.GetString
    (RawScenarioDescription, 1, 3)
243     End Function
244
245     Public Function ParseAuto_CAFE_STD(ByRef ActSheet As Excel.Worksheet) As Boolean
246         Dim i As Integer
247
248         ' resize the model years array for the current scenario
249         ReDim Me.Scenarios.Scenarios(ScenarioIdx).MY(Me.nMY - 1)
250
251         ' find min and max years
252         MinYear = Double.NaN
253         MaxYear = Double.NaN
254
255         ' first get model years
256         For i = 0 To Me.nMY - 1
257             Me.Scenarios.Scenarios(ScenarioIdx).MY(i) = GetDouble(RawAuto_CAFE_STD, 1, 3 +
    i)
258             ' find min and max years
259             If Double.IsNaN(MinYear) Then
260                 Me.MinYear = Me.Scenarios.Scenarios(ScenarioIdx).MY(i)
261             Else
262                 Me.MinYear = Math.Min(MinYear, Me.Scenarios.Scenarios(ScenarioIdx).MY(i))
263             End If
264
265             If Double.IsNaN(MaxYear) Then
266                 Me.MaxYear = Me.Scenarios.Scenarios(ScenarioIdx).MY(i)
267             Else
268                 Me.MaxYear = Math.Max(MaxYear, Me.Scenarios.Scenarios(ScenarioIdx).MY(i))
269             End If
270         Next i
271
```

```
272         ' save min and max years to ParametersClass
273         Me.Scenarios.Scenarios(ScenarioIdx).MinYear = Me.MinYear
274         Me.Scenarios.Scenarios(ScenarioIdx).MaxYear = Me.MaxYear
275
276         ReDim Me.Scenarios.Scenarios(ScenarioIdx).CafeFlatSTD(MaxYear - 2000)
277         'ReDim m_Parameters.Auto_CAFE_STD(Me.nMY - 1)
278
279         For i = 0 To Me.nMY - 1
280             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
281             ' then get auto cafe standards
282             Me.Scenarios.Add_CAFE_RegClass_FlatSTD(ScenarioIdx, m, Double.NaN)
                        ' RC 0
283             Me.Scenarios.Add_CAFE_RegClass_FlatSTD(ScenarioIdx, m, Me.GetDouble
    (RawAuto_CAFE_STD, 2, 3 + i))    ' RC 1 - domestic
284             Me.Scenarios.Add_CAFE_RegClass_FlatSTD(ScenarioIdx, m, Me.GetDouble
    (RawAuto_CAFE_STD, 2, 3 + i))    ' RC 2 - foreign
285             'm_Parameters.Auto_CAFE_STD(i) = GetDouble(RawAuto_CAFE_STD, 2, 3 + i)
286         Next i
287     End Function
288
289     Public Function ParseRawApplicabilityOfLTProgram(ByRef ActSheet As Excel.Worksheet) As
    Boolean
290         Dim i As Integer
291         ReDim Me.Scenarios.Scenarios(ScenarioIdx).LtDefinition(MaxYear - 2000)
292         ReDim Me.Scenarios.Scenarios(ScenarioIdx).HltInclusion(MaxYear - 2000)
293
294         For i = 0 To Me.nMY - 1
295             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
296             ' first get lt definition
297             Me.Scenarios.Scenarios(ScenarioIdx).LtDefinition(m) = Me.GetInteger
    (RawApplicabilityOfLTProgram, 2, 3 + i)
298             ' then get hlt inclusion
299             Me.Scenarios.Scenarios(ScenarioIdx).HltInclusion(m) = Me.GetInteger
    (RawApplicabilityOfLTProgram, 3, 3 + i)
300         Next i
301     End Function
302
303     Public Function ParseLT_RegClassBoundaries(ByRef ActSheet As Excel.Worksheet) As
    Boolean
304         Dim i, j As Integer
305         Dim nRCs As Integer
306         ReDim Me.Scenarios.Scenarios(ScenarioIdx).LtRegClassBoundaries(MaxYear - 2000)
307         ReDim Me.Scenarios.Scenarios(ScenarioIdx).nLtRegClass(MaxYear - 2000)
308
309         ' first get the attribute
310         Me.Scenarios.Scenarios(ScenarioIdx).LtRegClassAttribute = GetString
    (RawLT_RegClassBoundaries, 1, 3)
311         If UCase(Left(Me.Scenarios.Scenarios(ScenarioIdx).LtRegClassAttribute, 1)) = "A"
    Then
312             Me.Scenarios.Scenarios(ScenarioIdx).LtRegClassAttribute = "Area"
313         ElseIf UCase(Left(Me.Scenarios.Scenarios(ScenarioIdx).LtRegClassAttribute, 1)) = "W
    " Then
314             Me.Scenarios.Scenarios(ScenarioIdx).LtRegClassAttribute = "Weight"
315         Else
316             Me.Scenarios.Scenarios(ScenarioIdx).LtRegClassAttribute = "None"
317         End If
318         ' then get the reg class boundaries
319         For i = 0 To Me.nMY - 1
320             nRCs = 0
321             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
322
323             ' fill in the first 3 rows with nothing
324             Me.Scenarios.Add_LT_RegClass_Boundaries(ScenarioIdx, m, Double.NaN)
325             Me.Scenarios.Add_LT_RegClass_Boundaries(ScenarioIdx, m, Double.NaN)
326             Me.Scenarios.Add_LT_RegClass_Boundaries(ScenarioIdx, m, Double.NaN)
327
328             For j = 0 To Me.nLT_RegClass - 1
329                 Dim dbl As Double = Me.GetDouble(RawLT_RegClassBoundaries, 4 + j, 3 + i)
330
331                 If Not Double.IsNaN(dbl) Then
```

```
332                     nRCs += 1
333                     Me.Scenarios.Add_LT_RegClass_Boundaries(ScenarioIdx, m, dbl)
334                 End If
335             Next j
336
337             Me.Scenarios.Scenarios(ScenarioIdx).nLtRegClass(m) = nRCs
338         Next i
339     End Function
340
341     Public Function ParseLT_RegClassFlatSTD(ByRef ActSheet As Excel.Worksheet) As Boolean
342         Dim i, j As Integer
343         'ReDim m_Parameters.LT_RegClass_FlatSTD(Me.nLT_RegClass - 1, Me.nMY - 1)
344
345         For i = 0 To Me.nMY - 1
346             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
347
348             For j = 0 To Me.nLT_RegClass - 1
349                 Dim dbl As Double = Me.GetDouble(RawLT_RegClassFlatSTD, 2 + j, 3 + i)
350
351                 If Not Double.IsNaN(dbl) Then
352                     Me.Scenarios.Add_CAFE_RegClass_FlatSTD(ScenarioIdx, m, dbl)
353                 End If
354
355                 'Me.Scenarios.LT_RegClass_FlatSTD(i, j) = GetDouble(RawLT_RegClassFlatSTD, ↙
    2 + i, 3 + j)
356             Next j
357         Next i
358     End Function
359
360     Public Function ParseLT_Trade(ByRef ActSheet As Excel.Worksheet) As Boolean
361         Dim i As Integer
362         ReDim Me.Scenarios.Scenarios(ScenarioIdx).TradeAllowed(MaxYear - 2000)
363         ReDim Me.Scenarios.Scenarios(ScenarioIdx).TradeDiscount(MaxYear - 2000)
364
365         For i = 0 To Me.nMY - 1
366             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
367             ' first get trade allowed
368             Dim str As String = GetString(RawLT_Trade, 2, 3 + i)
369             If Not IsNothing(str) Then
370                 If str.ToLower() = "y" Then
371                     Me.Scenarios.Scenarios(ScenarioIdx).TradeAllowed(m) = True
372                 Else
373                     Me.Scenarios.Scenarios(ScenarioIdx).TradeAllowed(m) = False
374                 End If
375             Else
376                 Me.Scenarios.Scenarios(ScenarioIdx).TradeAllowed(m) = False
377             End If
378             ' then get trade discount
379             Me.Scenarios.Scenarios(ScenarioIdx).TradeDiscount(m) = Me.GetDouble(RawLT_Trade ↙
    , 3, 3 + i)
380         Next i
381     End Function
382
383     Public Function ParseLT_FunctionType(ByRef ActSheet As Excel.Worksheet) As Boolean
384         Dim i As Integer
385         ReDim Me.Scenarios.Scenarios(ScenarioIdx).LtFunctionType(MaxYear - 2000)
386
387         For i = 0 To Me.nMY - 1
388             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
389             Me.Scenarios.Scenarios(ScenarioIdx).LtFunctionType(m) = Me.GetInteger          ↙
    (RawLT_FunctionType, 2, 3 + i)
390         Next i
391     End Function
392
393     Public Function ParseLT_FunctionCoefficient(ByRef ActSheet As Excel.Worksheet) As      ↙
    Boolean
394         Dim i, j As Integer
395         ReDim Me.Scenarios.Scenarios(ScenarioIdx).LtFunctionCoefficients(MaxYear - 2000, Me ↙
    .nCoefficient - 1)
396
```

```
397         For i = 0 To Me.nMY - 1
398             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
399             For j = 0 To Me.nCoefficient - 1
400                 Me.Scenarios.Scenarios(ScenarioIdx).LtFunctionCoefficients(m, j) = _
401                                                     GetDouble(RawLT_FunctionCoefficient, 2 + j, 3 + i)
402             Next j
403         Next i
404     End Function
405
406     Public Function ParseHLT_FlatSTD(ByRef ActSheet As Excel.Worksheet) As Boolean
407         Dim i As Integer
408         ReDim Me.Scenarios.Scenarios(ScenarioIdx).HLT_FlatSTD(MaxYear - 2000)
409
410         For i = 0 To Me.nMY - 1
411             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
412             Me.Scenarios.Scenarios(ScenarioIdx).HLT_FlatSTD(m) = Me.GetDouble(RawHLT_FlatSTD, 2, 3 + i)
413         Next i
414     End Function
415     Public Function ParseTransitionalFlatSTD(ByRef ActSheet As Excel.Worksheet) As Boolean
416         Dim i As Integer
417         ReDim Me.Scenarios.Scenarios(ScenarioIdx).TransitionalFlatSTD(MaxYear - 2000)
418
419         For i = 0 To Me.nMY - 1
420             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
421             Me.Scenarios.Scenarios(ScenarioIdx).TransitionalFlatSTD(m) = Me.GetDouble(RawTransitionalFlatSTD, 2, 3 + i)
422         Next i
423     End Function
424
425     Public Function ParseCAFE_FineRate(ByRef ActSheet As Excel.Worksheet) As Boolean
426         Dim i As Integer
427         ReDim Me.Scenarios.Scenarios(ScenarioIdx).CafeFineRate(MaxYear - 2000)
428
429         For i = 0 To Me.nMY - 1
430             Dim m As Integer = Me.Scenarios.Scenarios(ScenarioIdx).MY(i) - 2000
431             Me.Scenarios.Scenarios(ScenarioIdx).CafeFineRate(m) = Me.GetDouble(RawCAFE_FineRate, 2, 3 + i)
432         Next i
433     End Function
434
435 #End Region
436
437
438     Private Function CheckField(ByRef ASheet As Excel.Worksheet, ByVal FieldName As String, _
439                                 ByRef RtnRange As Excel.Range, _
440                                 Optional ByVal PostException As Boolean = True) As Boolean
441         Try
442             RtnRange = ASheet.Range(FieldName)
443         Catch ex As Exception
444             If PostException Then
445                 MsgBox("Cannot find '" & FieldName & "' field in the Parameters file." & vbCr & vbCr & "Aborting", MsgBoxStyle.Critical Or MsgBoxStyle.OKOnly)
446             End If
447             CheckField = False
448             RtnRange = Nothing
449             Exit Function
450         End Try
451         CheckField = True
452     End Function
453
454     Private Sub FindMissing(ByVal DesiredTitle As String, ByVal ColNum As Integer, ByRef ErrorString As String)
455         If ColNum = 0 Then
456             If ErrorString = String.Empty Then
457                 ErrorString = DesiredTitle
458             Else
459                 ErrorString = ErrorString + NewLine + DesiredTitle
```

```
460             End If
461         End If
462     End Sub
463
464 End Class
465
```

```
Public Class RawTechnologies
    Inherits RawInput
    ' Reads the Technology file

#Region "Data Members"
    Public VehicleTechnology As VehicleTechnologiesClass
    Private Technologies As TechnologiesClass
    Private Technology As TechnologyClass
    Public NameColNum As Integer
    Public AbbrColNum As Integer
    Public FC_LowColNum As Integer
    Public FC_HighColNum As Integer
    Public Cost_LowColNum As Integer
    Public Cost_HighColNum As Integer
    Public YearAvailableColNum As Integer
    Public Path1ColNum As Integer
    Public Path2ColNum As Integer
    Public Path3ColNum As Integer
    Public kWeightColNum As Integer
    Public PhaseInColNum As Integer
    ''''' added for tech separation '''''
    Public TechTypeColNum As Integer
#End Region

    Public Sub New()
    End Sub

    Public Sub New(ByRef CForm As CafeForm, ByVal fName As String) ' With input file name
        Dim TechnologyFile As New C_Cafe_InFile
        Dim nSheets As Integer
        Dim nCol As Integer
        Dim nRow As Integer
        Dim s As Integer    ' for sheet indexing
        Dim c As Integer    ' for column indexing
        Dim r As Integer    ' for row indexing
        Dim ActSheet As Excel.Worksheet
        Dim Range0 As Excel.Range
        Dim Range1 As Excel.Range
        Dim SheetName As String
        Dim ExtraNRCClasses() As String
        Dim FoundNRCClasses(9) As Boolean
        Dim FoundNRCClassesStr() As String = {"SUV-Small", "SUV - Mid", "SUV - Large",
    "Minivan", _
                                              "Pickup-Small", "Pickup - Large", "Subcompact
    ", _
                                              "Compact", "Midsize", "Large" _
                                             }

        If Not TechnologyFile.OpenFile(fName, 4) Then
            ' could not open file for any reason, prompt user and exit...
            MsgBox("An error has occured while opening the technologies input file.  " & _
                   "Please try again.", MsgBoxStyle.Exclamation, "Error Opening File")
            Exit Sub
        End If

        Wb = TechnologyFile.GetWorkbookObject
        nSheets = Wb.Worksheets.Count

        ' parse wb title
        If Not Me.ParseWbTitle("CAFE Technology Description file", 1.0) Then
            ' if wb title or version did not match desired once
            ' close file and exit sub
            TechnologyFile.CloseFile()
            TechnologyFile = Nothing
            Exit Sub
        End If

        VehicleTechnology = New VehicleTechnologiesClass
        TechnologyFile.Activate()

```

```
69          For s = 0 To nSheets - 1
70              ActSheet = TechnologyFile.SetSheet(s + 1)
71              SheetName = ActSheet.Name.Trim()
72
73              Select Case UCase(SheetName)
74                  Case UCase(FoundNRCClassesStr(0))
75                      FoundNRCClasses(0) = True
76                      '
77                  Case UCase(FoundNRCClassesStr(1))
78                      FoundNRCClasses(1) = True
79                      '
80                  Case UCase(FoundNRCClassesStr(2))
81                      FoundNRCClasses(2) = True
82                      '
83                  Case UCase(FoundNRCClassesStr(3))
84                      FoundNRCClasses(3) = True
85                      '
86                  Case UCase(FoundNRCClassesStr(4))
87                      FoundNRCClasses(4) = True
88                      '
89                  Case UCase(FoundNRCClassesStr(5))
90                      FoundNRCClasses(5) = True
91                      '
92                  Case UCase(FoundNRCClassesStr(6))
93                      FoundNRCClasses(6) = True
94                      '
95                  Case UCase(FoundNRCClassesStr(7))
96                      FoundNRCClasses(7) = True
97                      '
98                  Case UCase(FoundNRCClassesStr(8))
99                      FoundNRCClasses(8) = True
100                     '
101                 Case UCase(FoundNRCClassesStr(9))
102                     FoundNRCClasses(9) = True
103                     '
104                 Case Else
105                     ' this sheet name is not on our list...make a note of it
106                     If IsNothing(ExtraNRCClasses) Then
107                         ReDim Preserve ExtraNRCClasses(0)
108                     Else
109                         ReDim Preserve ExtraNRCClasses(ExtraNRCClasses.Length)
110                     End If
111                     ExtraNRCClasses(ExtraNRCClasses.Length - 1) = SheetName
112             End Select
113         Next s
114
115         ' first check for all required worksheets
116         For s = 0 To FoundNRCClasses.Length - 1
117             If Not FoundNRCClasses(s) Then
118                 ' missing s'th worksheet
119                 MsgBox("The following ""NRC Class"" worksheet is missing: " & _
120                        FoundNRCClassesStr(s), MsgBoxStyle.Critical, "Missing worksheet")
121                 '
122                 ' close file and exit sub
123                 TechnologyFile.CloseFile()
124                 TechnologyFile = Nothing
125                 Exit Sub
126             End If
127         Next s
128
129         ' next check for any extra worksheets
130         If Not IsNothing(ExtraNRCClasses) Then
131             ' there are extra worksheets in this workbook...
132             Dim ExtraNRCStr As String = ""
133             For s = 0 To ExtraNRCClasses.Length - 1
134                 ExtraNRCStr += (ExtraNRCClasses(s) & ", ")
135             Next s
136             ExtraNRCStr = ExtraNRCStr.Substring(0, ExtraNRCStr.Length - 2)
137
138             MsgBox("This workbook contains the following undefined ""NRC Classes""
```

```
worksheet(s): " & _
                ExtraNRCStr, MsgBoxStyle.Information, "Extra worksheets")
            '
            ' close file and exit sub
            TechnologyFile.CloseFile()
            TechnologyFile = Nothing
            Exit Sub
        End If

        ' now we can load and check the data
        For s = 0 To nSheets - 1
            ActSheet = TechnologyFile.SetSheet(s + 1)
            SheetName = ActSheet.Name

            nCol = TechnologyFile.GetNumColumns(2)
            nRow = TechnologyFile.GetNumRows()

            ' Get the header
            RawHeaders = ActSheet.Range("2:2").Resize.Value

            ' Now get the data
            Range0 = ActSheet.Cells(3, 1)
            Range1 = ActSheet.Cells(nRow, nCol)
            RawData = ActSheet.Range(Range0, Range1).Resize.Value

            ' Determine the column numbers
            ' Fixed columns first
            NameColNum = 1
            For c = 2 To nCol
                CheckColumnName(RawHeaders, "FC-Low", 1, c, FC_LowColNum)
                CheckColumnName(RawHeaders, "FC-High", 1, c, FC_HighColNum)
                CheckColumnName(RawHeaders, "Cost-Low", 1, c, Cost_LowColNum)
                CheckColumnName(RawHeaders, "Cost-High", 1, c, Cost_HighColNum)
                CheckColumnName(RawHeaders, "Year Avail.", 1, c, YearAvailableColNum)
                CheckColumnName(RawHeaders, "Path1", 1, c, Path1ColNum)
                CheckColumnName(RawHeaders, "Path2", 1, c, Path2ColNum)
                CheckColumnName(RawHeaders, "Path3", 1, c, Path3ColNum)
                CheckColumnName(RawHeaders, "Abbr.", 1, c, AbbrColNum)
                CheckColumnName(RawHeaders, "Phase-in", 1, c, PhaseInColNum)
                CheckColumnName(RawHeaders, "kWeight", 1, c, kWeightColNum)
                CheckColumnName(RawHeaders, "TechType", 1, c, TechTypeColNum)    ' tech
separation
            Next c

            ' Check results
            Dim MsgString As String = ""

            Me.CheckRequiredColumn(Me.FC_LowColNum, "    - FC-Low", MsgString)
            Me.CheckRequiredColumn(Me.FC_HighColNum, "    - FC-High", MsgString)
            Me.CheckRequiredColumn(Me.Cost_LowColNum, "    - Cost-Low", MsgString)
            Me.CheckRequiredColumn(Me.Cost_HighColNum, "    - Cost-High", MsgString)
            Me.CheckRequiredColumn(Me.YearAvailableColNum, "    - Year Avail.", MsgString)
            Me.CheckRequiredColumn(Me.Path1ColNum, "    - Path1", MsgString)
            Me.CheckRequiredColumn(Me.Path2ColNum, "    - Path2", MsgString)
            Me.CheckRequiredColumn(Me.Path3ColNum, "    - Path3", MsgString)
            Me.CheckRequiredColumn(Me.AbbrColNum, "    - Abbr.", MsgString)
            Me.CheckRequiredColumn(Me.PhaseInColNum, "    - Phase-in", MsgString)
            Me.CheckRequiredColumn(Me.kWeightColNum, "    - kWeight", MsgString)
            Me.CheckRequiredColumn(Me.TechTypeColNum, "    - TechType", MsgString)  ' tech
separation

            If MsgString <> "" Then
                MsgBox("Could not find match for the following column(s): " & NewLine & _
                       NewLine & MsgString & NewLine & "Please select a different workbook.
", _
                       MsgBoxStyle.Exclamation, "Missing Columns")
                Exit Sub
            End If

            ' Copy the data...
```

```
205             Technologies = New TechnologiesClass(SheetName)
206             For r = 1 To nRow - 2
207                 Technology = New TechnologyClass
208                 Technology.Name = GetString(RawData, r, NameColNum)
209                 Technology.FC_Low = GetDouble(RawData, r, FC_LowColNum)
210                 Technology.FC_High = GetDouble(RawData, r, FC_HighColNum)
211                 Technology.Cost_Low = GetDouble(RawData, r, Cost_LowColNum)
212                 Technology.Cost_High = GetDouble(RawData, r, Cost_HighColNum)
213                 Technology.YearAvailable = GetInteger(RawData, r, YearAvailableColNum)
214                 Technology.Path1 = GetBoolean(RawData, r, Path1ColNum)
215                 Technology.Path2 = GetBoolean(RawData, r, Path2ColNum)
216                 Technology.Path3 = GetBoolean(RawData, r, Path3ColNum)
217                 Technology.kWeight = GetDouble(RawData, r, kWeightColNum)
218                 Technology.PhaseIn = GetDouble(RawData, r, PhaseInColNum)
219                 Technology.Abbr = GetString(RawData, r, AbbrColNum)
220                 Technology.TechType = GetString(RawData, r, TechTypeColNum)      ' tech
    separation
221
222                 ' Check for material substitution (has lots of special cases downstream)
223                 If Technology.TechType = "MSM" Or Technology.TechType = "MSP" Or Technology
    .TechType = "WR" Then
224                     Technology._IsMaterialSubstitution = True
225                     If Technology.Abbr = "MSX" Then
226                         Technology.Removable = True
227                     End If
228                 End If
229                 If Technology.Abbr.ToUpper = "DSL" Then
230                     Technology._IsDieselization = True
231                 End If
232                 Technologies.Add(Technology)
233                 Technology = Nothing
234             Next r
235             VehicleTechnology.Add(Technologies)
236             Technologies = Nothing
237         Next s
238
239         ' Create global instance of technologies and copy technologies to it
240         CAFE.Technologies = New VehicleTechnologiesClass
241         VehicleTechnologiesClass.Copy(CAFE.Technologies, Me.VehicleTechnology)
242
243         ' Close the file
244         TechnologyFile.CloseFile()
245         TechnologyFile = Nothing
246
247         MyBase.ValidFile = True
248     End Sub
249
250 End Class
251
```

```
Public Class RawTransmissionsClass
    Inherits RawInput .

#Region "TransmissionColumnNumbers"
    Private Transmission_CodeColNum As Integer
    Private ManufacturerColNum As Integer
    Private NameColNum As Integer
    Private OriginColNum As Integer
    Private TypeColNum As Integer
    Private NumberForwardGearsColNum As Integer
    Private ControlColNum As Integer
    Private LogicColNum As Integer
    Private GearRatioFirstGearColNum As Integer     '''''''''''''''
    Private GearRatioSecondGearColNum As Integer    '''''''''''''''
    Private GearRatioThirdGearColNum As Integer     '''''''''''''''
    Private GearRatioFourthGearColNum As Integer    '''''''''''''''
    Private GearRatioFifthGearColNum As Integer     '''''''''''''''
    Private GearRatioSixthGearColNum As Integer     '''''''''''''''
    Private ReverseGearRatioColNum As Integer       '''''''''''''''
    Private MaxGearRatioColNum As Integer
    Private MinGearRatioColNum As Integer
    Private TCRatioColNum As Integer
    Private AxleRatioColNum As Integer      '''''''''''''''
    Private TCLockupBypassColNum As Integer
    Private TransmissionFluidSpecificationColNum As Integer
    Private TransmissionFluidViscosityColNum As Integer
    Private TransmissionNotesColNum As Integer

    Public SP5ColNum As Integer     ' field for the 5SP col
    Public SP6ColNum As Integer     ' field for the 6SP col
    Public ASLColNum As Integer
    Public CVTColNum As Integer
    Public ASTColNum As Integer
    Public ACVTColNum As Integer
    Public HEVColNum As Integer
#End Region

    Public Sub New()
    End Sub

    Public Sub New(ByVal fName As String)
        ' Read transmissions file and store raw data
        Dim TransmissionsFile As New C_Cafe_InFile
        Dim RowCount As Integer
        Dim ColCount As Integer

        If Not TransmissionsFile.OpenFile(fName, 3) Then
            ' could not open file for any reason, prompt user and exit...
            MsgBox("An error has occured while opening the transmissions input file.  " & _
                   "Please try again.", MsgBoxStyle.Exclamation, "Error Opening File")
            Exit Sub
        End If

        RowCount = TransmissionsFile.GetNumRows()
        ColCount = TransmissionsFile.GetNumColumns(1)
        Wb = TransmissionsFile.GetWorkbookObject

        ' parse wb title
        If Not Me.ParseWbTitle("CAFE Transmissions Dataset file", 1.0) Then
            ' if wb title or version did not match desired ones
            ' close file and exit sub
            TransmissionsFile.CloseFile()
            TransmissionsFile = Nothing
            Exit Sub
        End If

        ' load the data
        Me.LoadData(wb.ActiveSheet, RowCount, ColCount)

        ' Close the file
```

```
71          TransmissionsFile.CloseFile()
72          TransmissionsFile = Nothing
73      End Sub
74
75      '''
76      ' Loads the data from the given worksheet.
77      '
78      Public Sub LoadData(ByVal ws As Excel.Worksheet, ByVal RowCount As Integer, ByVal
        ColCount As Integer)
79          Dim ActSheet As Excel.Worksheet
80          Dim rng0 As Excel.Range
81          Dim rng1 As Excel.Range
82          Dim i As Integer
83          Dim j As Integer
84          Dim r As Integer
85          Dim MsgString As String
86
87          ' Access the worksheet.
88          ActSheet = ws
89
90          ' Headers first
91          Me.RawHeaders = ActSheet.Range("1:1").Resize.Value
92
93          ' Now the data
94          rng0 = ActSheet.Cells(2, 1)
95          rng1 = ActSheet.Cells(RowCount, ColCount)
96          Me.RawData = ActSheet.Range(rng0, rng1).Resize.Value
97
98          ' Parse the column headers
99          ReDim ColMatch(ColCount)
100         For i = 1 To ColCount
101             CheckColumnName("Transmission Code", 1, i, Transmission_CodeColNum)
102             CheckColumnName("Manufacturer", 1, i, ManufacturerColNum)
103             CheckColumnName("Name", 1, i, NameColNum)
104             CheckColumnName("Origin", 1, i, OriginColNum)
105             CheckColumnName("Type", 1, i, TypeColNum)
106             CheckColumnName("Number of Forward Gears", 1, i, NumberForwardGearsColNum)
107             CheckColumnName("Control", 1, i, ControlColNum)
108             CheckColumnName("Logic", 1, i, LogicColNum)
109             CheckColumnName("Gear Ratio First Gear", 1, i, GearRatioFirstGearColNum)
110             CheckColumnName("Gear Ratio Second Gear", 1, i, GearRatioSecondGearColNum)
111             CheckColumnName("Gear Ratio Third Gear", 1, i, GearRatioThirdGearColNum)
112             CheckColumnName("Gear Ratio Fourth Gear", 1, i, GearRatioFourthGearColNum)
113             CheckColumnName("Gear Ratio Fifth Gear", 1, i, GearRatioFifthGearColNum)
114             CheckColumnName("Gear Ratio Sixth Gear", 1, i, GearRatioSixthGearColNum)
115             CheckColumnName("Reverse Gear Ratio", 1, i, ReverseGearRatioColNum)
116             CheckColumnName("Gear Ratio (Max.)", 1, i, MaxGearRatioColNum)
117             CheckColumnName("Gear Ratio (Min.)", 1, i, MinGearRatioColNum)
118             CheckColumnName("TC Ratio", 1, i, TCRatioColNum)
119             CheckColumnName("Axle Ratio", 1, i, AxleRatioColNum)
120             CheckColumnName("TC Lockup/Bypass", 1, i, TCLockupBypassColNum)
121             CheckColumnName("Transmission Fluid Specification", 1, i,
        TransmissionFluidSpecificationColNum)
122             CheckColumnName("Transmission Lubricant Viscosity", 1, i,
        TransmissionFluidViscosityColNum)
123             CheckColumnName("Transmission Notes", 1, i, TransmissionNotesColNum)
124             CheckColumnName("5SP", 1, i, SP5ColNum)
125             CheckColumnName("6SP", 1, i, SP6ColNum)
126             CheckColumnName("ASL", 1, i, ASLColNum)
127             CheckColumnName("CVT", 1, i, CVTColNum)
128             CheckColumnName("AST", 1, i, ASTColNum)
129             CheckColumnName("ACVT", 1, i, ACVTColNum)
130             CheckColumnName("HEV", 1, i, HEVColNum)
131         Next i
132
133         ' Check if the required columns are missing
134         MsgString = ""
135         Me.CheckRequiredColumn(Me.Transmission_CodeColNum, "    - Transmission Code",
        MsgString)
136         Me.CheckRequiredColumn(Me.ManufacturerColNum, "    - Manufacturer", MsgString)
```

```
137         Me.CheckRequiredColumn(Me.NameColNum, "    - Name", MsgString)
138         Me.CheckRequiredColumn(Me.TypeColNum, "    - Type", MsgString)
139         Me.CheckRequiredColumn(Me.NumberForwardGearsColNum, "    - Number of Forward Gears"
    , MsgString)
140         Me.CheckRequiredColumn(Me.ControlColNum, "    - Control", MsgString)
141         Me.CheckRequiredColumn(Me.LogicColNum, "    - Logic", MsgString)
142
143         ' Now check for any missing column names
144         'MsgString = ""
145         'For i = 1 To ColCount
146         '    If ColMatch(i) = False Then
147         '        MsgString = MsgString + i.ToString + " "
148         '    End If
149         'Next i
150
151         If MsgString <> "" Then
152             MsgBox("Could not find match for the following column(s): " & NewLine & _
153                    NewLine & MsgString & NewLine & "Please select a different workbook.", _
154                    MsgBoxStyle.Exclamation, "Missing Columns")
155             Exit Sub
156         End If
157
158         ' Now step through the engines and record the essentials...
159         ReDim CAFE.MasterTransmissions(RowCount - 2)
160         For r = 2 To RowCount
161             i = r - 2
162             j = r - 1
163             CAFE.MasterTransmissions(i) = New TransmissionsClass
164
165             Dim Valid As Boolean = True
166             Dim Reasons As String = String.Empty
167
168             Dim TransmissionCode As Integer = GetInteger(j, Transmission_CodeColNum)
169             Dim Manufacturer As String = GetString(j, ManufacturerColNum)
170             Dim Type As String = GetString(j, TypeColNum)
171             Dim Origin As String = GetString(j, OriginColNum)
172             Dim NumberForwardGears As Integer = GetInteger(j, NumberForwardGearsColNum)
173             Dim control As String = GetString(j, ControlColNum)
174
175             Valid = Valid And ValidateField(TransmissionCode, 0, Integer.MaxValue,
    "TransmissionCode", Reasons)
176             Valid = Valid And ValidateField(Manufacturer, "Manufacturer", Reasons)
177             Valid = Valid And ValidateField(Type, "Type", Reasons)
178             If Type.ToUpper = "C" Or Type.ToUpper = "T" Then
179                 ' If it's either clutch or torque-converter, then we should specify Control
 and numfwdgears
180                 Valid = Valid And ValidateField(NumberForwardGears, 3, Integer.MaxValue,
    "NumberForwardGears", Reasons)
181                 Valid = Valid And ValidateField(control, "Control", Reasons)
182             End If
183
184             CAFE.MasterTransmissions(i).Valid = Valid
185             CAFE.MasterTransmissions(i).ReasonForBeingInvalid = Reasons
186             CAFE.MasterTransmissions(i).MyIndexNumber = i
187             CAFE.MasterTransmissions(i).Transmission_Code = GetInteger(j,
    Transmission_CodeColNum)
188             CAFE.MasterTransmissions(i).Manufacturer = GetString(j, ManufacturerColNum)
189             CAFE.MasterTransmissions(i).Name = GetString(j, NameColNum)
190             CAFE.MasterTransmissions(i).Type = GetString(j, TypeColNum)
191             CAFE.MasterTransmissions(i).Origin = GetString(j, OriginColNum)
192             CAFE.MasterTransmissions(i).NumberForwardGears = GetInteger(j,
    NumberForwardGearsColNum)
193             CAFE.MasterTransmissions(i).Control = GetString(j, ControlColNum)
194             CAFE.MasterTransmissions(i).Logic = GetString(j, LogicColNum)
195             CAFE.MasterTransmissions(i).GearRatioFirstGear = GetDouble(j,
    GearRatioFirstGearColNum)
196             CAFE.MasterTransmissions(i).GearRatioSecondGear = GetDouble(j,
    GearRatioSecondGearColNum)
197             CAFE.MasterTransmissions(i).GearRatioThirdGear = GetDouble(j,
    GearRatioThirdGearColNum)
```

```
198             CAFE.MasterTransmissions(i).GearRatioFourthGear = GetDouble(j,
    GearRatioFourthGearColNum)
199             CAFE.MasterTransmissions(i).GearRatioFifthGear = GetDouble(j,
    GearRatioFifthGearColNum)
200             CAFE.MasterTransmissions(i).GearRatioSixthGear = GetDouble(j,
    GearRatioSixthGearColNum)
201             CAFE.MasterTransmissions(i).ReverseGearRatio = GetDouble(j,
    ReverseGearRatioColNum)
202             CAFE.MasterTransmissions(i).MaxGearRatio = GetDouble(j, MaxGearRatioColNum)
203             CAFE.MasterTransmissions(i).MinGearRatio = GetDouble(j, MinGearRatioColNum)
204             CAFE.MasterTransmissions(i).TCRatio = GetDouble(j, TCRatioColNum)
205             CAFE.MasterTransmissions(i).AxleRatio = GetDouble(j, AxleRatioColNum)
206             CAFE.MasterTransmissions(i).TCLockupBypass = GetBoolean(j,
    TCLockupBypassColNum)
207             CAFE.MasterTransmissions(i).TransmissionFluidSpecification = GetString(j,
    TransmissionFluidSpecificationColNum)
208             CAFE.MasterTransmissions(i).TransmissionFluidViscosity = GetString(j,
    TransmissionFluidViscosityColNum)
209             CAFE.MasterTransmissions(i).TransmissionNotes = GetString(j,
    TransmissionNotesColNum)
210             CAFE.MasterTransmissions(i).TechOver5SP = Me.GetTechOverRide(GetString(j,
    SP5ColNum))
211             CAFE.MasterTransmissions(i).TechOver6SP = Me.GetTechOverRide(GetString(j,
    SP6ColNum))
212             CAFE.MasterTransmissions(i).TechOverASL = Me.GetTechOverRide(GetString(j,
    ASLColNum))
213             CAFE.MasterTransmissions(i).TechOverCVT = Me.GetTechOverRide(GetString(j,
    CVTColNum))
214             CAFE.MasterTransmissions(i).TechOverAST = Me.GetTechOverRide(GetString(j,
    ASTColNum))
215             CAFE.MasterTransmissions(i).TechOverACVT = Me.GetTechOverRide(GetString(j,
    ACVTColNum))
216             CAFE.MasterTransmissions(i).TechOverHEV = Me.GetTechOverRide(GetString(j,
    HEVColNum))
217         Next r
218         MyBase.ValidFile = True
219     End Sub
220
221     '''
222     ' Validate string field
223     ' Simple check ...
224     ' If the field is blank, return FALSE and append string to FailureReason
225     Private Function ValidateField(ByVal field As String, _
226                     ByVal FieldName As String, _
227                     ByRef FailureReason As String) As Boolean
228         Dim Valid As Boolean = True
229         If field = String.Empty Then
230             Valid = False
231             FailureReason += FieldName + " is blank;"
232         End If
233         Return Valid
234     End Function
235
236     '''
237     ' Validate Double Field
238     ' Checks that the value is (a) a number, and (b) is between the specified min/max
239     Private Function ValidateField(ByVal field As Double, _
240                     ByVal Min As Double, ByVal Max As Double, _
241                     ByVal FieldName As String, _
242                     ByRef FailureReason As String) As Boolean
243         Dim valid = True
244         If field < Min Then
245             valid = False
246             FailureReason += FieldName + " is LT than min value of " + Min.ToString + ";"
247         ElseIf field > Max Then
248             valid = False
249             FailureReason += FieldName + " is GT than max value of " + Max.ToString + ";"
250         End If
251         Return valid
252     End Function
```

```
253
254     '''
255     ' Validate Integer Field
256     ' Checks that the value is (a) a number, and (b) is between the specified min/max
257     Private Function ValidateField(ByVal field As Integer, _
258                     ByVal Min As Integer, ByVal Max As Integer, _
259                     ByVal FieldName As String, _
260                     ByRef FailureReason As String) As Boolean
261         Dim valid = True
262         If field < Min Then
263             valid = False
264             FailureReason += FieldName + " is LT than min value of " + Min.ToString + ";"
265         ElseIf field > Max Then
266             valid = False
267             FailureReason += FieldName + " is GT than max value of " + Max.ToString + ";"
268         End If
269         Return valid
270     End Function
271
272     Private Function GetTechOverRide(ByVal TechOverRideField As Object) As CAFE.
        TechOverRide
273         Dim Str As String
274         Dim RtnCode As CAFE.TechOverRide
275         Str = GetString(TechOverRideField).ToUpper
276         If Str = "FORCE" Or Str = "F" Then
277             RtnCode = CAFE.TechOverRide.Force
278         ElseIf Str = "SKIP" Or Str = "S" Then
279             RtnCode = CAFE.TechOverRide.Skip
280         ElseIf Str = "USED" Or Str = "S" Then
281             RtnCode = CAFE.TechOverRide.Used
282         Else
283             RtnCode = CAFE.TechOverRide.None
284         End If
285         Return RtnCode
286     End Function
287 End Class
288
```

```
Public Class TransmissionsClass

#Region "TransmissionsData"
    Public Valid As Boolean  ' Run-time generated
    Public ReasonForBeingInvalid As String
    Public MyIndexNumber As Integer     ' Run-time generated

    Public Shared nMostModels As Integer = 0     ' Stores the number of models for the
    '                                            ' transmission with the most models

    Public Transmission_Code As Integer
    Public Manufacturer As String
    Public MfgIdx As Integer
    Public Name As String
    Public Origin As String
    Public Type As String
    Public NumberForwardGears As Integer
    Public Control As String
    Public Logic As String
    Public GearRatioFirstGear As Double      '''''''''''''''
    Public GearRatioSecondGear As Double     '''''''''''''''
    Public GearRatioThirdGear As Double      '''''''''''''''
    Public GearRatioFourthGear As Double     '''''''''''''''
    Public GearRatioFifthGear As Double      '''''''''''''''
    Public GearRatioSixthGear As Double      '''''''''''''''
    Public ReverseGearRatio As Double        '''''''''''''''
    Public MaxGearRatio As Double
    Public MinGearRatio As Double
    Public TCRatio As Double
    Public AxleRatio As Double       '''''''''''''''
    Public TCLockupBypass As Boolean
    Public TransmissionFluidSpecification As String
    Public TransmissionFluidViscosity As String
    Public TransmissionNotes As String

    Public TechOver5SP As CAFE.TechOverRide      ' field for the 5SP col
    Public TechOver6SP As CAFE.TechOverRide      ' field for the 6SP col
    Public TechOverASL As CAFE.TechOverRide
    Public TechOverCVT As CAFE.TechOverRide
    Public TechOverAST As CAFE.TechOverRide
    Public TechOverACVT As CAFE.TechOverRide
    Public TechOverHEV As CAFE.TechOverRide

    Public ParentIndex As Integer    ' Index to the parent transmission, if there is one

    Public nCompleteModels As Integer        ' Number of models in complete list
    Public CompleteModelsList() As Integer   ' Complete list of models which use this engine
    Public nModels As Integer                ' Number of valid models that use this
    transmission
    Public ModelsList() As Integer           ' List of valid models that use this
    transmission
    Public TechCost As Double                ' Technology cost applied to this transmission.
    Public TechImprv As Double               ' Technology improvements applied to this
    transmission.
    Public TechUsed() As Boolean             ' Transmission Techs used by this transmission
#End Region

    Public Sub New()
        ParentIndex = -1     ' set parent index to -1 to indicate that this
        '                    ' transmission does not have a parent transmission
        Valid = False
        nModels = 0
        TechCost = 0
        TechImprv = 1
        ReDim TechUsed(-1)
    End Sub
    Public Shared Function Clone(ByVal srcIdx As Integer, ByVal VehList() As Integer) As
    Integer
        Dim tIdx As Integer = CAFE.Transmissions.Length
        Debug.WriteLine("Adding new transmission (# " + tIdx.ToString + ")")
```

```
67
68          ' Check if the number of vehicles for this trans is the same as the 'most'
69          ' if so, we'll want to reset 'most' after our cloning.
70          Dim RecalcMost As Boolean = (CAFE.Transmissions(srcIdx).nCompleteModels =
        nMostModels)
71
72          ReDim Preserve CAFE.Transmissions(tIdx)
73          TransmissionsClass.Copy(CAFE.Transmissions(tIdx), CAFE.Transmissions(srcIdx))
74          ' Now remove vehicles being split off from source engine
75          ' Do CompleteModelsList
76          For v As Integer = 0 To VehList.Length - 1
77              Dim vv As Integer = 0
78              Do
79                  If VehList(v) = CAFE.Transmissions(srcIdx).CompleteModelsList(vv) Then
80                      ' Remove this vehicle
81                      Dim r As Integer
82                      For r = vv + 1 To CAFE.Transmissions(srcIdx).CompleteModelsList.Length
        - 1
83                          CAFE.Transmissions(srcIdx).CompleteModelsList(r - 1) = CAFE.
        Transmissions(srcIdx).CompleteModelsList(r)
84                      Next
85                      r = CAFE.Transmissions(srcIdx).CompleteModelsList.Length - 2
86                      ReDim Preserve CAFE.Transmissions(srcIdx).CompleteModelsList(r)
87                      CAFE.Transmissions(srcIdx).nCompleteModels = r + 1
88                  Else
89                      vv = vv + 1
90                  End If
91              Loop While vv < CAFE.Transmissions(srcIdx).CompleteModelsList.Length
92              ' Add the vehicle to the new transmission
93              CAFE.Transmissions(tIdx).AddUser(VehList(v))
94          Next v
95          ' Now populate ModelsList for both
96          CAFE.Transmissions(srcIdx).CreateModelsList()
97          CAFE.Transmissions(tIdx).CreateModelsList()
98
99          If RecalcMost Then
100             nMostModels = 0
101             For t As Integer = 0 To CAFE.Transmissions.Length - 1
102                 nMostModels = Math.Max(nMostModels, CAFE.Transmissions(t).nCompleteModels)
103             Next
104         End If
105         Return tIdx
106     End Function
107
108     Public Sub Merge(ByVal FromTransmissionIdx As Integer)
109         ' Merge an engine... assumes that the caller knows that the two engines are
110         ' indentical or not.
111         For Each vidx As Integer In CAFE.Transmissions(FromTransmissionIdx).ModelsList
112             Me.AddUser(vidx)
113         Next
114     End Sub
115     Public Sub Copy(ByRef Dst As TransmissionsClass)
116         TransmissionsClass.Copy(Dst, Me)
117     End Sub
118     Public Shared Sub Copy(ByRef Dst As TransmissionsClass, ByRef Src As
        TransmissionsClass)
119         Dim i As Integer
120
121         Dst = New TransmissionsClass
122
123         Dst.Valid = Src.Valid
124         Dst.MyIndexNumber = Src.MyIndexNumber
125         Dst.ParentIndex = Src.ParentIndex
126
127         Dst.Transmission_Code = Src.Transmission_Code
128         Dst.Manufacturer = Src.Manufacturer
129         Dst.Name = Src.Name
130         Dst.Type = Src.Type
131         Dst.Origin = Src.Origin
132         Dst.NumberForwardGears = Src.NumberForwardGears
```

```
133         Dst.Control = Src.Control
134         Dst.Logic = Src.Logic
135         Dst.GearRatioFirstGear = Src.GearRatioFirstGear
136         Dst.GearRatioSecondGear = Src.GearRatioSecondGear
137         Dst.GearRatioThirdGear = Src.GearRatioThirdGear
138         Dst.GearRatioFourthGear = Src.GearRatioFourthGear
139         Dst.GearRatioFifthGear = Src.GearRatioFifthGear
140         Dst.GearRatioSixthGear = Src.GearRatioSixthGear
141         Dst.ReverseGearRatio = Src.ReverseGearRatio
142         Dst.MaxGearRatio = Src.MaxGearRatio
143         Dst.MinGearRatio = Src.MinGearRatio
144         Dst.TCRatio = Src.TCRatio
145         Dst.AxleRatio = Src.AxleRatio
146         Dst.TCLockupBypass = Src.TCLockupBypass
147         Dst.TransmissionFluidSpecification = Src.TransmissionFluidSpecification
148         Dst.TransmissionFluidViscosity = Src.TransmissionFluidViscosity
149         Dst.TransmissionNotes = Src.TransmissionNotes
150         Dst.TechOver5SP = Src.TechOver5SP
151         Dst.TechOver6SP = Src.TechOver6SP
152         Dst.TechOverASL = Src.TechOverASL
153         Dst.TechOverCVT = Src.TechOverCVT
154         Dst.TechOverAST = Src.TechOverAST
155         Dst.TechOverACVT = Src.TechOverACVT
156         Dst.TechOverHEV = Src.TechOverHEV
157
158         Dst.nCompleteModels = Src.nCompleteModels
159         ReDim Dst.CompleteModelsList(Dst.nCompleteModels - 1)
160         For i = 0 To Dst.nCompleteModels - 1
161             Dst.CompleteModelsList(i) = Src.CompleteModelsList(i)
162         Next
163
164         Dst.nModels = Src.nModels
165         ReDim Dst.ModelsList(Src.nModels - 1)
166         For i = 0 To Src.nModels - 1
167             Dst.ModelsList(i) = Src.ModelsList(i)
168         Next
169         Dst.TechCost = Src.TechCost
170         Dst.TechImprv = Src.TechImprv
171         Dst.MfgIdx = Src.MfgIdx
172         If IsNothing(Src.TechUsed) = False Then
173             ReDim Dst.TechUsed(Src.TechUsed.Length - 1)
174             For i = 0 To Src.TechUsed.Length - 1
175                 Dst.TechUsed(i) = Src.TechUsed(i)
176             Next
177         End If
178     End Sub
179     Public Sub AddUser(ByVal VehIndex As Integer)
180         ' Add a vehicle (index to vehicles) as a vehicle which uses this transmission
181         ReDim Preserve CompleteModelsList(nCompleteModels)      ' need to preserve the
    array each time  <ms>
182         CompleteModelsList(nCompleteModels) = VehIndex
183         nCompleteModels = nCompleteModels + 1
184
185         If nCompleteModels > nMostModels Then
186             nMostModels = nCompleteModels
187         End If
188     End Sub
189
190     '''
191     ' CreateModelsList
192     ' This sub populates the ModelsList from valid vehicles in the CompleteModelsList
193     '
194     Public Sub CreateModelsList()
195         nModels = 0
196         ReDim ModelsList(-1)
197         If Me.Valid Then
198             For v As Integer = 0 To CompleteModelsList.Length - 1
199                 If CAFE.Vehicles(CompleteModelsList(v)).IsValid Then
200                     ' got a valid one
201                     ReDim Preserve ModelsList(ModelsList.Length)
```

```
202                     ModelsList(ModelsList.Length - 1) = CompleteModelsList(v)
203                 End If
204             Next
205             nModels = ModelsList.Length
206             Dim NumTechnologies = CAFE.Technologies.GetNumTechnologies
207         End If
208     End Sub
209
210     '''
211     ' Clears the CompleteModelsList() and ModelsList() arrays for the new run.
212     '
213     Public Sub ClearModelLists()
214         ReDim Me.CompleteModelsList(-1)
215         Me.nCompleteModels = 0
216         ReDim Me.ModelsList(-1)
217         Me.nModels = 0
218     End Sub
219
220 #Region "TechOverRide Support Routines"
221     Public Function IsTechOverRideForce(ByVal TechIdx As Integer) As Boolean
222         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Force
223     End Function
224     Public Function IsTechOverRideForce(ByVal TechAbbreviation As String) As Boolean
225         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Force
226     End Function
227     Public Function IsTechOverRideSkip(ByVal TechIdx As Integer) As Boolean
228         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Skip
229     End Function
230     Public Function IsTechOverRideSkip(ByVal TechAbbreviation As String) As Boolean
231         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Skip
232     End Function
233     Public Function IsTechOverRideUsed(ByVal TechIdx As Integer) As Boolean
234         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Used
235     End Function
236     Public Function IsTechOverRideUsed(ByVal TechAbbreviation As String) As Boolean
237         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Used
238     End Function
239     Public Function IsTechOverRideNone(ByVal TechIdx As Integer) As Boolean
240         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.None
241     End Function
242     Public Function IsTechOverRideNone(ByVal TechAbbreviation As String) As Boolean
243         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.None
244     End Function
245     '''
246     ' Get the CAFE.TechOverRIde setting.
247     ' This is rather brute-force with a complete lack of elegance,
248     ' maybe we'll clean this up someday -- but don't count on it
249     ' jvs 15 oct 04
250     Public Function GetTechOverRide(ByVal TechIdx As Integer) As CAFE.TechOverRide
251         Dim Abbr As String
252         ' Get the abbreviation using NRC Vehicle Class 0 (the abbrv are the same for all)
253         Abbr = CAFE.Technologies.VehicleClasses(0).Technology(TechIdx).Abbr
254         Return GetTechOverRide(Abbr)
255     End Function
256     Public Function GetTechOverRide(ByVal TechAbbreviation As String) As CAFE.TechOverRide
257         Dim RtnCode As CAFE.TechOverRide = CAFE.TechOverRide.None
258         Select Case TechAbbreviation
259             Case "5SP"
260                 RtnCode = Me.TechOver5SP
261             Case "6SP"
262                 RtnCode = Me.TechOver6SP
263             Case "ASL"
264                 RtnCode = Me.TechOverASL
265             Case "CVT"
266                 RtnCode = Me.TechOverCVT
267             Case "AST"
268                 RtnCode = Me.TechOverAST
269             Case "ACVT"
270                 RtnCode = Me.TechOverACVT
271             Case "HEV"
```

```
272                 RtnCode = Me.TechOverHEV
273         End Select
274         Return RtnCode
275     End Function
276 #End Region
277 End Class
278
```

```
Public Class RawVehiclesClass
    Inherits RawInput

#Region "ColumnNumbers"
    ' Column numbers of header categories (from line 1 of header)
    Protected GeneralColNum As Integer
    Protected PlanningColNum As Integer
    Protected SalesColNum As Integer
    Protected SalesColCount As Integer          ' jvs 17 june 04
    Protected MSRPColNum As Integer
    Protected MSRPColCount As Integer           ' jvs 17 june 04
    Protected PriceColNum As Integer
    Protected PriceColCount As Integer          ' jvs 17 june 04
    Protected HybridizationColNum As Integer
    Protected VehicleColNum As Integer
    Protected RegulatoryColNum As Integer
    Protected LTDefinitionColNum As Integer     '''''''''''''''
    Protected HLTInclusionColNum As Integer     '''''''''''''''
    Protected EffectsColNum As Integer          '''''''''''''''
    Protected TechnologyApplicabilityOverridesColNum As Integer     '''''''''''''''
    Protected StageAnalysisFEColStart As Integer
    Protected StageAnalysisFEColCount As Integer
    Protected StageAnalysisCWColStart As Integer
    Protected StageAnalysisCWColCount As Integer
    Protected StageAnalysisSalesColStart As Integer
    Protected StageAnalysisSalesColCount As Integer

    ' Column numbers of specific columns
    Protected NumberColNum As Integer
    Protected ManufacturerColNum As Integer
    Protected ModelColNum As Integer
    Protected NamePlateColNum As Integer        '''''''''''''''
    Protected FuelEconomyColNum As Integer
    Protected ActualFE_FFVsColNum As Integer    '''''''''''''''
    Protected EngineCodeColNum As Integer   ' Index to engine type
    Protected TransmissionCodeColNum As Integer
    Protected OriginColNum As Integer
    Protected US_Canadian_ContentColNum As Integer
    'Protected FinalAssemblyCityColNum As Integer
    'Protected FinalAssemblyStateColNum As Integer
    'Protected FinalAssemblyColNum As Integer
    'Protected GeneralNotesColNum As Integer
    Protected PredecessorColNum As Integer
    Protected LastFreshening ColNum As Integer
    Protected NextFreshening ColNum As Integer
    Protected LastRedesignColNum As Integer
    Protected NextRedesignColNum As Integer
    Protected EmploymentHoursPerVehicleColNum As Integer    '''''''''''''''
    'Protected PlanningNotesColNum As Integer
    Protected nSalesYears As Integer
    Protected SalesYearsColNums() As Integer
    Protected SalesYears() As Integer
    'Protected SalesNotesColNum As Integer
    Protected nMSRPYears As Integer
    Protected MSRPYearsColNums() As Integer
    Protected MSRPYears() As Integer
    'Protected MSRPNotesColNum As Integer
    Protected nPriceYears As Integer
    Protected PriceYearsColNums() As Integer
    Protected PriceYears() As Integer
    'Protected PriceNotesColNum As Integer
    Protected HybridTypeColNum As Integer
    Protected VoltageOrPressureColNum As Integer
    Protected EnergyStorageCapacityColNum As Integer
    Protected BatteryTypeColNum As Integer
    Protected EnergyTransferColNum As Integer
    Protected BrakingEnergyRecoveryColNum As Integer
    Protected ShareMaximumPowerColNum As Integer
    'Protected HybridizationNotesColNum As Integer
    Protected StyleColNum As Integer
```

```
71      Protected VehClassColNum As Integer
72      Protected VehStructureColNum As Integer
73      Protected DriveColNum As Integer
74      Protected FinalDriveRatioColNum As Integer
75      Protected EngineSpeedOverVehSpeedColNum As Integer
76      'Protected FrontAxleViscosityColNum As Integer
77      'Protected RearAxleViscosityColNum As Integer
78      'Protected OverAllLengthColNum As Integer '[KG:  changed 01/24/05 for footprint-based
        system]
79      'Protected OverAllWidthColNum As Integer '[KG:  changed 01/24/05 for footprint-based
        system]
80      Protected OverAllHeightColNum As Integer
81      Protected WheelbaseColNum As Integer '[KG:  changed 01/24/05 for footprint-based
        system]
82      Protected TrackWidthFrontColNum As Integer '[KG:  changed 01/24/05 for footprint-based
         system]
83      Protected TrackWidthRearColNum As Integer '[KG:  changed 01/24/05 for footprint-based
        system]
84      'Protected GroundClearanceColNum As Integer
85      'Protected FrontAxleClearanceColNum As Integer
86      'Protected RearAxleClearanceColNum As Integer
87      'Protected AngleOfApproachColNum As Integer
88      'Protected BreakoverAngleColNum As Integer
89      'Protected AngleOfDepartureColNum As Integer
90      'Protected CenterOfGravityHeightColNum As Integer
91      Protected CurbWeightColNum As Integer
92      Protected TestWeightColNum As Integer
93      Protected PAUSettingColNum As Integer
94      Protected GVWRColNum As Integer
95      Protected StandardTowingCapacityColNum As Integer
96      Protected MaxTowingCapacityColNum As Integer
97      Protected PayloadColNum As Integer
98      Protected MinSeatingColNum As Integer
99      Protected MaxSeatingColNum As Integer
100     'Protected FrontRowSeatingColNum As Integer
101     'Protected CargoVolBehindFirstRowColNum As Integer
102     'Protected SecondRowSeatingColNum As Integer
103     'Protected SecondRowFlatCapabilityColNum As Integer
104     'Protected CargoVolBehindSecondRowColNum As Integer
105     'Protected ThirdRowSeatingColNum As Integer
106     'Protected ThirdRowFlatCapabilityColNum As Integer
107     'Protected CargoVolBehindThirdRowColNum As Integer
108     'Protected EnclosedVolumeColNum As Integer
109     'Protected StandardPassengerVolumeColNum As Integer
110     'Protected MaxPassengerVolumeColNum As Integer
111     'Protected CargoVolumeIndexColNum As Integer
112     'Protected StandardEnclosedCargoVolumeColNum As Integer  '''''''''''''''
113     'Protected MaxEnclosedCargoVolumeColNum As Integer       '''''''''''''''
114     'Protected OpenBoxAreaLengthColNum As Integer
115     'Protected MinOpenBoxAreaWidthColNum As Integer
116     'Protected MaxOpenBoxAreaWidthColNum As Integer
117     Protected OpenBoxAreaColNum As Integer
118     Protected OpenBoxHeightColNum As Integer
119     Protected FuelCapacityColNum As Integer
120     'Protected TireRollingResistanceColNum As Integer
121     'Protected FrontalAreaColNum As Integer
122     'Protected RecommendedTirePressureColNum As Integer      '''''''''''''''
123     'Protected TireStaticLoadedRadiusColNum As Integer       '''''''''''''''
124     'Protected TireTreadWidthColNum As Integer               '''''''''''''''
125     'Protected TireDeflectionColNum As Integer               '''''''''''''''
126     'Protected AerodynamicDragCoefficientColNum As Integer
127     'Protected VehicleNotesColNum As Integer
128     'Protected EPA_ClassColNum As Integer
129     'Protected EPA_CertificationColNum As Integer
130     'Protected LEV_ClassColNum As Integer
131     'Protected RegulatoryNotesColNum As Integer
132     Protected StageAnalysisFEColIsMY(-1) As Integer
133     Protected StageAnalysisCWColIsMY(-1) As Integer
134     Protected StageAnalysisSalesColIsMY(-1) As Integer
135
```

```
136     Protected LTDFN1ColNum As Integer
137     Protected LTDFN2ColNum As Integer
138     Protected LTDFN3ColNum As Integer
139     Protected LTDFN4ColNum As Integer
140     Protected LTDFN5ColNum As Integer
141     Protected HLTINCL1ColNum As Integer
142     Protected HLTINCL2ColNum As Integer
143     Protected HLTINCL3ColNum As Integer
144     Protected HLTINCL4ColNum As Integer
145     Protected HLTINCL5ColNum As Integer
146     'Protected MOBILE6ClassColNum As Integer
147     Protected SafetyClassColNum As Integer
148     Protected Percent2DRColNum As Integer
149     Protected MarketSegment_VOLPEColNum As Integer
150     Protected MarketSegment_AutoNewsColNum As Integer
151     'Protected MarketNestColNum As Integer      ' replaced by market segment
152     Protected ROLLColNum As Integer
153     Protected LDBColNum As Integer
154     Protected FADColNum As Integer
155     Protected EPSColNum As Integer
156     Protected EAIColNum As Integer
157     Protected AERColNum As Integer
158     Protected V42ColNum As Integer      ' the 42V column
159     Protected HEVColNum As Integer
160     Protected ISGColNum As Integer
161     Protected MS1ColNum As Integer
162     Protected MS2ColNum As Integer
163     Protected MS3ColNum As Integer
164     Protected MSxColNum As Integer
165 #End Region
166
167     ' Only processing is to organize information about the header columns
168     Public Sub New()
169     End Sub
170
171     Public Sub New(ByVal fName As String)    ' With input file name
172         ' Read vehicles file and store raw data
173         Dim VehiclesFile As New C_Cafe_InFile
174         Dim RowCount As Integer
175         Dim ColCount As Integer
176
177         If Not VehiclesFile.OpenFile(fName, 1) Then
178             ' could not open file for any reason, prompt user and exit...
179             MsgBox("An error has occured while opening the vehicles input file.  " & _
180                    "Please try again.", MsgBoxStyle.Exclamation, "Error Opening File")
181             Exit Sub
182         End If
183
184         RowCount = VehiclesFile.GetNumRows()
185         ColCount = VehiclesFile.GetNumColumns(2, True)
186         Wb = VehiclesFile.GetWorkbookObject
187
188         ' parse wb title
189         If Not Me.ParseWbTitle("CAFE Vehicle Dataset file", 1.0) Then
190             ' if wb title or version did not match desired once
191             ' close file and exit sub
192             VehiclesFile.CloseFile()
193             VehiclesFile = Nothing
194             Exit Sub
195         End If
196
197         ' load the data
198         Me.LoadData(Wb.ActiveSheet, RowCount, ColCount)
199
200         ' Close models file.
201         VehiclesFile.CloseFile()
202         VehiclesFile = Nothing
203     End Sub ' new RawVehiclesClass
204
205     '''
```

```
206     ' Loads the data from the given worksheet.
207     '
208     Public Sub LoadData(ByVal ws As Excel.Worksheet, ByVal RowCount As Integer, ByVal
        ColCount As Integer)
209         Dim ActSheet As Excel.Worksheet
210         Dim rng0 As Excel.Range
211         Dim rng1 As Excel.Range
212         Dim i As Integer
213         Dim j As Integer
214         Dim k As Integer
215         Dim r As Integer
216         Dim MsgString As String
217
218         ' Access the worksheet
219         ActSheet = ws
220
221         ' Headers first
222         Me.RawHeaders = ActSheet.Range("1:2").Resize.Value
223
224         ' Now the data
225         rng0 = ActSheet.Cells(3, 1)
226         rng1 = ActSheet.Cells(RowCount, ColCount)
227         Me.RawData = ActSheet.Range(rng0, rng1).Resize.Value
228
229         ' Now disect the data
230         ' Find categories in the top line of the header
231         Dim header As String
232         For i = 1 To ColCount
233             header = Me.RawHeaders(1, i)
234
235             If IsNothing(header) = False Then
236                 If header.ToUpper() = "General".ToUpper() Then
237                     GeneralColNum = i
238                 ElseIf header.ToUpper() = "Planning".ToUpper() Then
239                     PlanningColNum = i
240                 ElseIf header.ToUpper() = "Sales".ToUpper() Then
241                     SalesColNum = i
242                     SalesColCount = 1
243                     Do While IsNothing(rawheaders(1, i + SalesColCount))
244                         SalesColCount += 1
245                     Loop
246                 ElseIf header.ToUpper() = "MSRP".ToUpper() Then
247                     MSRPColNum = i
248                     MSRPColCount = 1
249                     Do While IsNothing(rawheaders(1, i + MSRPColCount))
250                         MSRPColCount += 1
251                     Loop
252                 ElseIf header.ToUpper() = "Price".ToUpper() Then
253                     PriceColNum = i
254                     PriceColCount = 1
255                     Do While IsNothing(rawheaders(1, i + PriceColCount))
256                         PriceColCount += 1
257                     Loop
258                 ElseIf header.ToUpper() = "STAGE ANALYSIS FE" Then
259                     StageAnalysisFEColStart = i
260                     StageAnalysisFEColCount = 0
261                     Do While IsNothing(rawheaders(2, i + StageAnalysisFEColCount)) = False
        _
262                         AndAlso Left(rawheaders(2, i + StageAnalysisFEColCount), 2) =
        "MY" _
263                         AndAlso IsNothing(rawheaders(1, i + StageAnalysisFEColCount +
        1))
264                         StageAnalysisFEColCount += 1
265                     Loop
266                 ElseIf header.ToUpper() = "STAGE ANALYSIS CURB WEIGHT" Then
267                     StageAnalysisCWColStart = i
268                     StageAnalysisCWColCount = 0
269                     Do While IsNothing(rawheaders(2, i + StageAnalysisCWColCount)) = False
        _
270                         AndAlso Left(rawheaders(2, i + StageAnalysisCWColCount), 2) =
```

```
"MY" _
                        AndAlso IsNothing(rawheaders(1, i + StageAnalysisCWColCount + 1))
                    StageAnalysisCWColCount += 1
                Loop
            ElseIf header.ToUpper() = "STAGE ANALYSIS SALES" Then
                StageAnalysisSalesColStart = i
                StageAnalysisSalesColCount = 0
                Do While IsNothing(rawheaders(2, i + StageAnalysisSalesColCount)) = False _
                        AndAlso Left(rawheaders(2, i + StageAnalysisSalesColCount), 2) = "MY" _
                        AndAlso IsNothing(rawheaders(1, i + StageAnalysisSalesColCount + 1))
                    StageAnalysisSalesColCount += 1
                Loop
            ElseIf header.ToUpper() = "Hybridization".ToUpper() Then
                HybridizationColNum = i
            ElseIf header.ToUpper() = "Vehicle".ToUpper() Then
                VehicleColNum = i
            ElseIf header.ToUpper() = "Regulatory".ToUpper() Or header = "Emissions".ToUpper() Then
                RegulatoryColNum = i
            ElseIf header.ToUpper() = "LT Definition".ToUpper() Then
                LTDefinitionColNum = i
            ElseIf header.ToUpper() = "HLT Inclusion".ToUpper() Then
                HLTInclusionColNum = i
            ElseIf header.ToUpper() = "Effects".ToUpper() Then
                EffectsColNum = i
            ElseIf header.ToUpper() = "Technology Applicability Overrides".ToUpper() Then
                TechnologyApplicabilityOverridesColNum = i
            End If
        End If
    Next i
    ' Now check that all the categories are found
    i = 0
    CheckCategories(GeneralColNum, "General", i, MsgString)
    CheckCategories(PlanningColNum, "Planning", i, MsgString)
    CheckCategories(SalesColNum, "Sales", i, MsgString)
    CheckCategories(MSRPColNum, "MSRP", i, MsgString)
    CheckCategories(PriceColNum, "Price", i, MsgString)
    CheckCategories(HybridizationColNum, "Hybridization", i, MsgString)
    CheckCategories(VehicleColNum, "Vehicle", i, MsgString)
    CheckCategories(RegulatoryColNum, "Regulatory or Emissions", i, MsgString)
    CheckCategories(LTDefinitionColNum, "LT Definition", i, MsgString)
    CheckCategories(HLTInclusionColNum, "HLT Inclusion", i, MsgString)
    CheckCategories(EffectsColNum, "Effects", i, MsgString)
    CheckCategories(TechnologyApplicabilityOverridesColNum, "Technology Applicability Overrides", i, MsgString)

    If i = 1 Then
        MsgBox("The following category is missing from the column headers: " & _
               NewLine & NewLine & MsgString & NewLine & _
               "Please select a different workbook.", _
               MsgBoxStyle.Exclamation, "Error In Input File")
        Return
    ElseIf i > 1 Then
        MsgBox("The following categories are missing from the column headers: " & _
               NewLine & NewLine & MsgString & NewLine & _
               "Please select a different workbook.", _
               MsgBoxStyle.Exclamation, "Error In Input File")
        Return
    End If

    ' Now parse through looking for the needed categories...
    ' This is long and tedious.  Perhaps doing this using sorted lists might be
    ' the way to go (but not yet).
    ReDim ColMatch(ColCount + 1)

```

```
333         For i = RawHeaders.GetLowerBound(0) To ColCount
334             CheckColumnName("Number", 2, i, NumberColNum)
335             CheckColumnName("Manufacturer", 2, i, ManufacturerColNum)
336             CheckColumnName("Model", 2, i, ModelColNum)
337             CheckColumnName("NamePlate", 2, i, NamePlateColNum)
338             CheckColumnName("Fuel Economy", 2, i, FuelEconomyColNum)
339             CheckColumnName("Actual FE (FFVs)", 2, i, ActualFE_FFVsColNum)
340             CheckColumnName("Engine Code", 2, i, EngineCodeColNum)
341             CheckColumnName("Transmission Code", 2, i, TransmissionCodeColNum)
342             CheckColumnName("Origin", 2, i, OriginColNum)
343             CheckColumnName("U.S./Canadian Content", 2, i, US_Canadian_ContentColNum)
344             'CheckColumnName("Final Assembly City", 2, i, FinalAssemblyCityColNum)
345             'CheckColumnName("Final Assembly State", 2, i, FinalAssemblyStateColNum)
346             'CheckColumnName("Final Assembly", 2, i, FinalAssemblyColNum)
347             'CheckColumnName("Final Assembly Country", 2, i, FinalAssemblyColNum)    ' same ↙
    col as above
348             'CheckColumnName("General Notes", 2, i, GeneralNotesColNum)
349             CheckColumnName("Predecessor", 2, i, PredecessorColNum)
350             CheckColumnName("Last Freshening", 2, i, LastFresheningColNum)
351             CheckColumnName("Next Freshening", 2, i, NextFresheningColNum)
352             CheckColumnName("Last Redesign", 2, i, LastRedesignColNum)
353             CheckColumnName("Next Redesign", 2, i, NextRedesignColNum)
354             CheckColumnName("Employment Hours Per Vehicle", 2, i,                    ↙
    EmploymentHoursPerVehicleColNum)
355             'CheckColumnName("Planning Notes", 2, i, PlanningNotesColNum)
356             'CheckColumnName("Sales Notes", 2, i, SalesNotesColNum)
357             'CheckColumnName("MSRP Notes", 2, i, MSRPNotesColNum)
358             'CheckColumnName("Price Notes", 2, i, PriceNotesColNum)
359             CheckColumnName("Type", 2, i, HybridTypeColNum)
360             CheckColumnName("Voltage (or Pressure)", 2, i, VoltageOrPressureColNum)
361             CheckColumnName("Energy Storage Capacity", 2, i, EnergyStorageCapacityColNum)
362             CheckColumnName("Battery Type", 2, i, BatteryTypeColNum)
363             CheckColumnName("Energy Transfer", 2, i, EnergyTransferColNum)
364             CheckColumnName("Braking Energy Recovery", 2, i, BrakingEnergyRecoveryColNum)
365             CheckColumnName("Share of Maximum Power", 2, i, ShareMaximumPowerColNum)
366             'CheckColumnName("Hybridizaton Notes", 2, i, HybridizationNotesColNum)
367             CheckColumnName("Style", 2, i, StyleColNum)
368             CheckColumnName("Class", 2, i, VehClassColNum)
369             CheckColumnName("Structure", 2, i, VehStructureColNum)
370             CheckColumnName("Drive", 2, i, DriveColNum)
371             CheckColumnName("Final Drive Ratio", 2, i, FinalDriveRatioColNum)
372             CheckColumnName("N/V", 2, i, EngineSpeedOverVehSpeedColNum)
373             'CheckColumnName("Front Axle Lubricant Viscosity", 2, i,                  ↙
    FrontAxleViscosityColNum)
374             'CheckColumnName("Rear Axle Lubricant Viscosity", 2, i,                   ↙
    RearAxleViscosityColNum)
375             'CheckColumnName("Overall Length", 2, i, OverAllLengthColNum) '[KG:  changed ↙
    01/24/05 for footprint-based system]
376             'CheckColumnName("Overall Width", 2, i, OverAllWidthColNum) '[KG:  changed 01/ ↙
    24/05 for footprint-based system]
377             'CheckColumnName("Overall Height", 2, i, OverAllHeightColNum)
378             CheckColumnName("Wheelbase", 2, i, WheelbaseColNum) '[KG:  changed 01/24/05    ↙
    for footprint-based system]
379             CheckColumnName("Track Width (front)", 2, i, TrackWidthFrontColNum) '[KG:      ↙
    changed 01/24/05 for footprint-based system]
380             CheckColumnName("Track Width (rear)", 2, i, TrackWidthRearColNum) '[KG:        ↙
    changed 01/24/05 for footprint-based system]
381             'CheckColumnName("Ground Clearance", 2, i, GroundClearanceColNum)
382             'CheckColumnName("Front Axle Clearance", 2, i, FrontAxleClearanceColNum)
383             'CheckColumnName("Rear Axle Clearance", 2, i, RearAxleClearanceColNum)
384             'CheckColumnName("Angle of Approach", 2, i, AngleOfApproachColNum)
385             'CheckColumnName("Breakover Angle", 2, i, BreakoverAngleColNum)
386             'CheckColumnName("Angle of Departure", 2, i, AngleOfDepartureColNum)
387             'CheckColumnName("Height of Center of Gravity", 2, i,                     ↙
    CenterOfGravityHeightColNum)
388             CheckColumnName("Curb Weight", 2, i, CurbWeightColNum)
389             CheckColumnName("Test Weight", 2, i, TestWeightColNum)
390             CheckColumnName("PAU Setting", 2, i, PAUSettingColNum)
391             CheckColumnName("GVWR", 2, i, GVWRColNum)
392             CheckColumnName("Towing Capacity (standard)", 2, i,                      ↙
```

```
StandardTowingCapacityColNum)
393             CheckColumnName("Towing Capacity (max)", 2, i, MaxTowingCapacityColNum)
394             CheckColumnName("Payload", 2, i, PayloadColNum)
395             CheckColumnName("Seating (min)", 2, i, MinSeatingColNum)
396             CheckColumnName("Seating (max)", 2, i, MaxSeatingColNum)
397             'CheckColumnName("Seating in First Row", 2, i, FrontRowSeatingColNum)
398             'CheckColumnName("Cargo Vol. Behind First Row", 2, i,
CargoVolBehindFirstRowColNum)
399             'CheckColumnName("Seating in Second Row", 2, i, SecondRowSeatingColNum)
400             'CheckColumnName("Second Row Flat Capability", 2, i,
SecondRowFlatCapabilityColNum)
401             'CheckColumnName("Cargo Vol. Behind Second Row", 2, i,
CargoVolBehindSecondRowColNum)
402             'CheckColumnName("Seating in Third Row", 2, i, ThirdRowSeatingColNum)
403             'CheckColumnName("Third Row Flat Capability", 2, i,
ThirdRowFlatCapabilityColNum)
404             'CheckColumnName("Cargo Vol. Behind Third Row", 2, i,
CargoVolBehindThirdRowColNum)
405             'CheckColumnName("Enclosed Volume", 2, i, EnclosedVolumeColNum)
406             'CheckColumnName("Passenger Volume (standard)", 2, i,
StandardPassengerVolumeColNum)
407             'CheckColumnName("Passenger Volume (max)", 2, i, MaxPassengerVolumeColNum)
408             'CheckColumnName("Cargo Volume Index", 2, i, CargoVolumeIndexColNum)
409             'CheckColumnName("Enclosed Cargo Volume (standard)", 2, i,
StandardEnclosedCargoVolumeColNum)
410             'CheckColumnName("Enclosed Cargo Volume (max)", 2, i,
MaxEnclosedCargoVolumeColNum)
411             'CheckColumnName("Open Box Area Length", 2, i, OpenBoxAreaLengthColNum)
412             'CheckColumnName("Open Box Area Width (standard)", 2, i,
MinOpenBoxAreaWidthColNum)
413             'CheckColumnName("Open Box Area Width (max)", 2, i, MaxOpenBoxAreaWidthColNum)
414             CheckColumnName("Open Box Area", 2, i, OpenBoxAreaColNum)
415             CheckColumnName("Open Box Height", 2, i, OpenBoxHeightColNum)
416             CheckColumnName("Fuel Capacity", 2, i, FuelCapacityColNum)
417             'CheckColumnName("Tire Rolling Resistance", 2, i, TireRollingResistanceColNum)
418             'CheckColumnName("Frontal Area", 2, i, FrontalAreaColNum)
419             'CheckColumnName("Recommended Tire Pressure", 2, i,
RecommendedTirePressureColNum)
420             'CheckColumnName("Tire Static Loaded Radius", 2, i,
TireStaticLoadedRadiusColNum)
421             'CheckColumnName("Tire Tread Width", 2, i, TireTreadWidthColNum)
422             'CheckColumnName("Tire Deflection", 2, i, TireDeflectionColNum)
423             'CheckColumnName("Aerodynamic Drag Coefficient", 2, i,
AerodynamicDragCoefficientColNum)
424             'CheckColumnName("Vehicle Notes", 2, i, VehicleNotesColNum)
425             'CheckColumnName("EPA Class", 2, i, EPA_ClassColNum)
426             'CheckColumnName("EPA Certification Bin", 2, i, EPA_CertificationColNum)
427             'CheckColumnName("LEV Class", 2, i, LEV_ClassColNum)
428             'CheckColumnName("Regulatory Notes", 2, i, RegulatoryNotesColNum)
429
430             CheckColumnName("LTDFN1", 2, i, LTDFN1ColNum)
431             CheckColumnName("LTDFN2", 2, i, LTDFN2ColNum)
432             CheckColumnName("LTDFN3", 2, i, LTDFN3ColNum)
433             CheckColumnName("LTDFN4", 2, i, LTDFN4ColNum)
434             CheckColumnName("LTDFN5", 2, i, LTDFN5ColNum)
435             CheckColumnName("HLTINCL1", 2, i, HLTINCL1ColNum)
436             CheckColumnName("HLTINCL2", 2, i, HLTINCL2ColNum)
437             CheckColumnName("HLTINCL3", 2, i, HLTINCL3ColNum)
438             CheckColumnName("HLTINCL4", 2, i, HLTINCL4ColNum)
439             CheckColumnName("HLTINCL5", 2, i, HLTINCL5ColNum)
440             'CheckColumnName("MOBILE6 Class", 2, i, MOBILE6ClassColNum)
441             CheckColumnName("Safety Class", 2, i, SafetyClassColNum)
442             CheckColumnName("% 2 DR Cars", 2, i, Percent2DRColNum)
443             CheckColumnName("Market Segment - VOLPE", 2, i, MarketSegment_VOLPEColNum)
444             CheckColumnName("Market Segment - Auto News", 2, i,
MarketSegment_AutoNewsColNum)
445             'CheckColumnName("Market Nest", 2, i, MarketNestColNum)        ' replaced by
market segment
446             CheckColumnName("ROLL", 2, i, ROLLColNum)
447             CheckColumnName("LDB", 2, i, LDBColNum)
```

```
448             CheckColumnName("FAD", 2, i, FADColNum)
449             CheckColumnName("EPS", 2, i, EPSColNum)
450             CheckColumnName("EAI", 2, i, EAIColNum)
451             CheckColumnName("AER", 2, i, AERColNum)
452             CheckColumnName("42V", 2, i, V42ColNum)
453             CheckColumnName("HEV", 2, i, HEVColNum)
454             CheckColumnName("ISG", 2, i, ISGColNum)
455             CheckColumnName("MS1", 2, i, MS1ColNum)
456             CheckColumnName("MS2", 2, i, MS2ColNum)
457             CheckColumnName("MS3", 2, i, MS3ColNum)
458             CheckColumnName("MSX", 2, i, MSxColNum)
459         Next i
460         ' jvs 17 june 04: Changed around how we step through SALES, MSRP and Price.
461         ' No assumption made about where the 'Notes' column is (or if it exists).
462
463         ' check if safety col and %2dr col are present and set whether to run safety class ↙
        calcs
464         CAFE.SafetyClassesLoaded = ((SafetyClassColNum <> 0) And (Percent2DRColNum <> 0))
465
466         ' Now parse through the Sales, MSRP, and Price fields
467         nSalesYears = 0
468         For i = SalesColNum To SalesColNum + SalesColCount - 1
469             If IsNothing(Me.RawHeaders(2, i)) = True Then
470                 MsgBox("Column " & i & ", under the Sales category, is missing a header." ↙
        & _
471                         "  Please double check your workbook and try again.", _
472                         MsgBoxStyle.Exclamation, "Missing Column Header")
473                 Exit Sub
474             End If
475             Dim tmpstring As String = rawheaders(2, i)
476             If tmpstring.Substring(0, 2) = "MY" Then
477                 ReDim Preserve SalesYearsColNums(nSalesYears)
478                 ReDim Preserve SalesYears(nSalesYears)
479                 SalesYearsColNums(nSalesYears) = i
480                 MsgString = Me.RawHeaders(2, i)
481                 SalesYears(nSalesYears) = CInt(MsgString.Substring(2))
482                 ColMatch(i) = True
483                 nSalesYears += 1
484             End If
485         Next i
486         '
487         nMSRPYears = 0
488         For i = MSRPColNum To MSRPColNum + MSRPColCount - 1
489             If IsNothing(Me.RawHeaders(2, i)) = True Then
490                 MsgBox("Column " & i & ", under the msrp category, is missing a header." & ↙
        _
491                         "  Please double check your workbook and try again.", _
492                         MsgBoxStyle.Exclamation, "Missing Column Header")
493                 Exit Sub
494             End If
495             Dim tmpstring As String = rawheaders(2, i)
496             If tmpstring.Substring(0, 2) = "MY" Then
497                 ReDim Preserve MSRPYearsColNums(nMSRPYears)
498                 ReDim Preserve MSRPYears(nMSRPYears)
499                 MSRPYearsColNums(nMSRPYears) = i
500                 MsgString = Me.RawHeaders(2, i)
501                 MSRPYears(nMSRPYears) = CInt(MsgString.Substring(2))
502                 ColMatch(i) = True
503                 nMSRPYears += 1
504             End If
505         Next i
506         '
507         nPriceYears = 0
508         For i = PriceColNum To PriceColNum + PriceColCount - 1
509             If IsNothing(Me.RawHeaders(2, i)) = True Then
510                 MsgBox("Column " & i & ", under the price category, is missing a header." ↙
        & _
511                         "  Please double check your workbook and try again.", _
512                         MsgBoxStyle.Exclamation, "Missing Column Header")
513                 Exit Sub
```

```
514             End If
515             Dim tmpstring As String = rawheaders(2, i)
516             If tmpstring.Substring(0, 2) = "MY" Then
517                 ReDim Preserve PriceYearsColNums(nPriceYears)
518                 ReDim Preserve PriceYears(nPriceYears)
519                 PriceYearsColNums(nPriceYears) = i
520                 MsgString = Me.RawHeaders(2, i)
521                 PriceYears(nPriceYears) = CInt(MsgString.Substring(2))
522                 ColMatch(i) = True
523                 nPriceYears += 1
524             End If
525         Next i
526
527         ' Check if the required columns are missing
528         MsgString = ""
529         Me.CheckRequiredColumn(Me.ManufacturerColNum, "    - Manufacturer", MsgString)
530         Me.CheckRequiredColumn(Me.ModelColNum, "    - Model", MsgString)
531         Me.CheckRequiredColumn(Me.NamePlateColNum, "    - NamePlate", MsgString)
532         Me.CheckRequiredColumn(Me.FuelEconomyColNum, "    - Fuel Economy", MsgString)
533         Me.CheckRequiredColumn(Me.ActualFE_FFVsColNum, "    - Actual FE (FFVs)",
    MsgString)
534         Me.CheckRequiredColumn(Me.EngineCodeColNum, "    - Engine Code", MsgString)
535         Me.CheckRequiredColumn(Me.TransmissionCodeColNum, "    - Transmission Code",
    MsgString)
536         Me.CheckRequiredColumn(Me.OriginColNum, "    - Origin", MsgString)
537         Me.CheckRequiredColumn(Me.StyleColNum, "    - Style", MsgString)
538         Me.CheckRequiredColumn(Me.VehClassColNum, "    - Class", MsgString)
539         'Me.CheckRequiredColumn(Me.OverAllLengthColNum, "    - Overall Length", MsgString)
     '[KG: changed 01/24/05 for footprint-based system]
540         'Me.CheckRequiredColumn(Me.OverAllWidthColNum, "    - Overall Width", MsgString) '
    [KG: changed 01/24/05 for footprint-based system]
541         Me.CheckRequiredColumn(Me.WheelbaseColNum, "    - Wheelbase", MsgString) '[KG:
    changed 01/24/05 for footprint-based system]
542         Me.CheckRequiredColumn(Me.TrackWidthFrontColNum, "    - Track Width (front)",
    MsgString) '[KG: changed 01/24/05 for footprint-based system]
543         Me.CheckRequiredColumn(Me.TrackWidthRearColNum, "    - Track Width (rear)",
    MsgString) '[KG: changed 01/24/05 for footprint-based system]
544         Me.CheckRequiredColumn(Me.TestWeightColNum, "    - Test Weight", MsgString)
545         Me.CheckRequiredColumn(Me.CurbWeightColNum, "    - Curb Weight", MsgString)
546         ' check that there is at least one col of Sales and MSRP
547         ' no sales years
548         Me.CheckRequiredColumn(nSalesYears, "    - Sales Model Years", MsgString)
549         ' no msrp years
550         Me.CheckRequiredColumn(nMSRPYears, "    - MSRP Model Years", MsgString)
551
552         '' Now check for any missing column names
553         'MsgString = ""
554         'For i = 1 To ColCount - 1
555         '    If ColMatch(i) = False Then
556         '        MsgString = MsgString + i.ToString & " "
557         '    End If
558         'Next i
559
560         If MsgString <> "" Then
561             MsgBox("Could not find match for the following column(s): " & NewLine &
562                    NewLine & MsgString & NewLine & "Please select a different workbook.",
    _
563                    MsgBoxStyle.Exclamation, "Missing Columns")
564             Exit Sub
565         End If
566
567         ' Moved MY scanning to here (09-mar-04 jvs)
568         ' Fill in ModelYears list
569         Dim myS As ArrayList = New ArrayList
570         Dim MaxSalesMY As Integer = Integer.MinValue
571         Dim mySales(nSalesYears - 1) As Integer
572         Dim tmpint As Integer
573         Dim tmpstr As String
574         For i = 0 To nSalesYears - 1
575             tmpstr = RawHeaders(2, Me.SalesYearsColNums(i)).ToString().Trim("MY".
```

```
    ToCharArray())
576             tmpint = CInt(tmpstr)
577             myS.Add(tmpint)
578             mySales(i) = tmpint
579             MaxSalesMY = Math.Max(tmpint, MaxSalesMY)
580         Next i
581         ' Sort the list
582         myS.Sort()
583
584         ' Now scan the MSRP years
585         Dim myM As ArrayList = New ArrayList
586         Dim MaxMsrpMY As Integer = Integer.MinValue
587         Dim myMSRP(nMSRPYears - 1) As Integer
588         For i = 0 To nMSRPYears - 1
589             tmpstr = RawHeaders(2, Me.MSRPYearsColNums(i)).ToString().Trim("MY".
    ToCharArray())
590             tmpint = CInt(tmpstr)
591             myM.Add(tmpint)
592             myMSRP(i) = tmpint
593             MaxMsrpMY = Math.Max(tmpint, MaxMsrpMY)
594         Next i
595         '
596         ' Scan the Price years
597         Dim myP As ArrayList = New ArrayList
598         Dim MaxPriceMY As Integer = Integer.MinValue
599         Dim myPrice(nPriceYears - 1) As Integer
600         For i = 0 To nPriceYears - 1
601             tmpstr = RawHeaders(2, Me.PriceYearsColNums(i)).ToString().Trim("MY".
    ToCharArray())
602             tmpint = CInt(tmpstr)
603             myP.Add(tmpint)
604             myPrice(i) = tmpint
605             MaxPriceMY = Math.Max(tmpint, MaxPriceMY)
606         Next i
607
608         ' Now compare myS and myM ... leaving MyS with the common set.
609         Dim MaxMY As Integer = Integer.MinValue
610         For Each tmpint In myS
611             If myM.Contains(tmpint) = False Then
612                 myS.Remove(tmpint)
613             ElseIf tmpint > MaxMY Then
614                 MaxMY = tmpint
615             End If
616         Next tmpint
617
618         ' Next compare myS and myP ... leaving MyS with the common set.
619         For Each tmpint In myS
620             If myP.Contains(tmpint) = False Then
621                 myS.Remove(tmpint)
622             ElseIf tmpint > MaxMY Then
623                 MaxMY = tmpint
624             End If
625         Next tmpint
626
627         ' Allocate the proper number of years in CAFE.ModelYearsSelected,
628         ' then 'turn on' the years which are present.
629         ReDim CAFE.ModelYearsSelected(CAFE.GMYI(MaxMY))
630         For Each tmpint In myS
631             CAFE.ModelYearsSelected(CAFE.GMYI(tmpint)) = True
632         Next
633         myM = Nothing
634         myS = Nothing
635         ' end of 09-mar-04 jvs
636
637         ' Now scan the Staging Analysis FE area to identify MYs
638         Dim MaxStageFE As Integer = Integer.MinValue
639         If StageAnalysisFEColCount > 0 Then
640             ReDim Me.StageAnalysisFEColIsMY(StageAnalysisFEColCount - 1)
641             ' grab the stage analysis fuel economy values
642             For i = StageAnalysisFEColStart To StageAnalysisFEColStart +
```

```
    StageAnalysisFEColCount - 1
643                 MsgString = Me.RawHeaders(2, i)
644                 StageAnalysisFEColIsMY(i - StageAnalysisFEColStart) = CInt(MsgString.Trim ↙
    ("MY".ToCharArray))
645                 MaxStageFE = Math.Max(MaxStageFE, StageAnalysisFEColIsMY(i - ↙
    StageAnalysisFEColStart))
646             Next
647         End If
648         '
649         ' Now scan the Staging Analysis curb weight area to identify MYs
650         Dim MaxStageCW As Integer = Integer.MinValue
651         If StageAnalysisCWColCount > 0 Then
652             ReDim Me.StageAnalysisCWColIsMY(StageAnalysisCWColCount - 1)
653             ' grab the stage analysis fuel economy values
654             For i = StageAnalysisCWColStart To StageAnalysisCWColStart + ↙
    StageAnalysisCWColCount - 1
655                 MsgString = Me.RawHeaders(2, i)
656                 StageAnalysisCWColIsMY(i - StageAnalysisCWColStart) = CInt(MsgString.Trim ↙
    ("MY".ToCharArray))
657                 MaxStageCW = Math.Max(MaxStageCW, StageAnalysisCWColIsMY(i - ↙
    StageAnalysisCWColStart))
658             Next
659         End If
660         ' Now scan the Staging Analysis sales area to identify MYs
661         Dim MaxStageSales As Integer = Integer.MinValue
662         If StageAnalysisSalesColCount > 0 Then
663             ReDim Me.StageAnalysisSalesColIsMY(StageAnalysisSalesColCount - 1)
664             ' grab the stage analysis fuel economy values
665             For i = StageAnalysisSalesColStart To StageAnalysisSalesColStart + ↙
    StageAnalysisSalesColCount - 1
666                 MsgString = Me.RawHeaders(2, i)
667                 StageAnalysisSalesColIsMY(i - StageAnalysisSalesColStart) = CInt(MsgString ↙
    .Trim("MY".ToCharArray))
668                 MaxStageSales = Math.Max(MaxStageSales, StageAnalysisSalesColIsMY(i - ↙
    StageAnalysisSalesColStart))
669             Next
670         End If
671
672
673         ' Now step through the vehicles and record their essentials...
674         ReDim CAFE.MasterVehicles(RowCount - 3)
675
676         For r = 3 To RowCount
677             i = r - 3
678             j = r - 2
679
680             ' Validate the entry before we accept it.
681             Dim ReasonsForBeingInvalid As String = String.Empty
682             Dim Valid As Boolean = True
683             Dim Manufacturer As String = GetString(j, ManufacturerColNum)
684             Dim Model As String = GetString(j, ModelColNum)
685             Dim NamePlate As String = GetString(j, NamePlateColNum)
686             Dim CodedFuelEconomy As Double = GetDouble(j, FuelEconomyColNum)
687             Dim ActualFE_FFVs As Double = GetDouble(j, ActualFE_FFVsColNum)
688             Dim EngineCode As Integer = GetInteger(j, EngineCodeColNum)
689             Dim TransmissionCode As Integer = GetInteger(j, TransmissionCodeColNum)
690             Dim Origin As String = GetString(j, OriginColNum)
691             Dim Style As String = GetString(j, StyleColNum)
692             Dim VehClass As String = GetString(j, VehClassColNum)
693             Dim CurbWeight As Double = GetDouble(j, CurbWeightColNum)
694             Dim TestWeight As Double = GetDouble(j, TestWeightColNum)
695
696             Valid = ValidateField(Manufacturer, "Manufacturer", ReasonsForBeingInvalid)
697             Valid = Valid And ValidateField(Model, "Model", ReasonsForBeingInvalid)
698             Valid = Valid And ValidateField(NamePlate, "NamePlate", ↙
    ReasonsForBeingInvalid)
699             Valid = Valid And ValidateField(CodedFuelEconomy, 0, Double.PositiveInfinity, ↙
    "Fuel Economy", ReasonsForBeingInvalid)
700             Valid = Valid And ValidateField(EngineCode, 0, Integer.MaxValue, "EngineCode", ↙
     ReasonsForBeingInvalid)
```

```
701             Valid = Valid And ValidateField(TransmissionCode, 0, Integer.MaxValue,
    "TransmissionCode", ReasonsForBeingInvalid)
702             Valid = Valid And ValidateField(Origin, "Origin", ReasonsForBeingInvalid)
703             Valid = Valid And ValidateField(Style, "Style", ReasonsForBeingInvalid)
704             Valid = Valid And ValidateField(VehClass, "VehClass", ReasonsForBeingInvalid)
705             Valid = Valid And ValidateField(CurbWeight, 0, Double.PositiveInfinity, "Curb
    Weight", ReasonsForBeingInvalid)
706             'ValidateField(TestWeight, 0, Double.PositiveInfinity, "Test Weight",
    ReasonsForBeingInvalid)
707
708             ' Now validate Sales, MSRP and Price.
709             ' There must be at least one year with an entry for all
710             Dim TempSales As Double
711             Dim TempMSRP As Double
712             Dim TempPrice As Double
713             Dim FoundMY As Boolean = False
714             For imy As Integer = 0 To Math.Min(nSalesYears, Math.Min(nMSRPYears,
    nPriceYears))
715                 TempSales = GetDouble(j, SalesColNum + imy)
716                 TempMSRP = GetDouble(j, MSRPColNum + imy)
717                 TempPrice = GetDouble(j, PriceColNum + imy)
718                 If TempSales > 0 And TempMSRP > 0 And TempPrice > 0 Then
719                     FoundMY = True
720                     Exit For
721                 End If
722             Next imy
723             If FoundMY = False Then
724                 Valid = False
725                 ReasonsForBeingInvalid += "Sales; MSRP; and/or Price not set;"
726             End If
727
728             CAFE.MasterVehicles(i) = New VehiclesClass
729
730             If Valid Then
731                 CAFE.MasterVehicles(i).ValidOverAll = True
732                 CAFE.MasterVehicles(i).ReasonForBeingInvalid = String.Empty
733             Else
734                 CAFE.MasterVehicles(i).ValidOverAll = False
735                 CAFE.MasterVehicles(i).ReasonForBeingInvalid = ReasonsForBeingInvalid
736             End If
737
738             CAFE.MasterVehicles(i).MyIndexNumber = i
739             CAFE.MasterVehicles(i).Number = GetInteger(j, NumberColNum)
740             CAFE.MasterVehicles(i).Manufacturer = Manufacturer
741             CAFE.MasterVehicles(i).Model = Model
742             CAFE.MasterVehicles(i).NamePlate = NamePlate
743             CAFE.MasterVehicles(i).CodedFuelEconomy = CodedFuelEconomy
744             CAFE.MasterVehicles(i).ActualFE_FFVs = ActualFE_FFVs
745             CAFE.MasterVehicles(i).InitialFuelEconomy = CAFE.MasterVehicles(i).FuelEconomy
746             CAFE.MasterVehicles(i).EngineCode = EngineCode
747             CAFE.MasterVehicles(i).TransmissionCode = TransmissionCode
748             CAFE.MasterVehicles(i).Origin = Origin
749             CAFE.MasterVehicles(i).US_Canadian_Content = GetDouble(j,
    US_Canadian_ContentColNum)
750             'CAFE.MasterVehicles(i).FinalAssemblyCity = GetString(j,
    FinalAssemblyCityColNum)
751             'CAFE.MasterVehicles(i).FinalAssemblyState = GetString(j,
    FinalAssemblyStateColNum)
752             'CAFE.MasterVehicles(i).FinalAssembly = GetString(j, FinalAssemblyColNum)
753             'CAFE.MasterVehicles(i).GeneralNotes = GetString(j, GeneralNotesColNum)
754             CAFE.MasterVehicles(i).PredecessorFromFile = GetInteger(j, PredecessorColNum)
755             CAFE.MasterVehicles(i).LastFreshening = GetInteger(j, LastFresheningColNum)
756             CAFE.MasterVehicles(i).NextFreshening = GetInteger(j, NextFresheningColNum)
757             CAFE.MasterVehicles(i).LastRedesign = GetInteger(j, LastRedesignColNum)
758             CAFE.MasterVehicles(i).NextRedesign = GetInteger(j, NextRedesignColNum)
759             CAFE.MasterVehicles(i).EmploymentHoursPerVehicle = GetDouble(j,
    EmploymentHoursPerVehicleColNum)
760             'CAFE.MasterVehicles(i).PlanningNotes = GetString(j, PlanningNotesColNum)
761
762             ' Allocate space for model years from 2000 -- 20xx (xx being max MY for Sales)
```

```
763             ' (09-mar-04 jvs)
764             ReDim CAFE.MasterVehicles(i).Sales(CAFE.GMYI(MaxSalesMY))
765             ReDim CAFE.MasterVehicles(i).InitialSales(CAFE.GMYI(MaxSalesMY))
766             ReDim CAFE.MasterVehicles(i).ModeledBaselineSales(CAFE.GMYI(MaxSalesMY)) ' KG: added 08/06/04
767             ReDim CAFE.MasterVehicles(i).TechCostByMY(CAFE.GMYI(MaxSalesMY))         ' jvs: added 12/11/05
768             CAFE.MasterVehicles(i).nSalesYears = CAFE.GMYI(MaxSalesMY) + 1
769             For k = 0 To nSalesYears - 1
770                 ' Copy sales values into proper spot in MY array
771                 CAFE.MasterVehicles(i).Sales(CAFE.GMYI(mySales(k))) = GetDouble(j, SalesColNum + k)
772                 CAFE.MasterVehicles(i).InitialSales(CAFE.GMYI(mySales(k))) = CAFE.MasterVehicles(i).Sales(CAFE.GMYI(mySales(k)))
773             Next k
774             ' Likewise for MSRP, allocate for all from 2000 -- 20yy .
775             ReDim CAFE.MasterVehicles(i).MSRP(CAFE.GMYI(MaxMsrpMY))
776             ReDim CAFE.MasterVehicles(i).InitialMSRP(CAFE.GMYI(MaxMsrpMY))
777             CAFE.MasterVehicles(i).nMSRPYears = CAFE.GMYI(MaxMsrpMY) + 1
778             For k = 0 To nMSRPYears - 1
779                 CAFE.MasterVehicles(i).MSRP(CAFE.GMYI(myMSRP(k))) = GetDouble(j, MSRPColNum + k)
780                 CAFE.MasterVehicles(i).InitialMSRP(CAFE.GMYI(myMSRP(k))) = CAFE.MasterVehicles(i).MSRP(CAFE.GMYI(myMSRP(k)))
781             Next k
782             ' (end of 09-mar-04 jvs)
783
784             ReDim CAFE.MasterVehicles(i).Price(CAFE.GMYI(MaxPriceMY))
785             ReDim CAFE.MasterVehicles(i).InitialPrice(CAFE.GMYI(MaxPriceMY))
786             CAFE.MasterVehicles(i).nPriceYears = CAFE.GMYI(MaxPriceMY) + 1
787             For k = 0 To nPriceYears - 1
788                 CAFE.MasterVehicles(i).Price(CAFE.GMYI(myPrice(k))) = GetDouble(j, PriceColNum + k)
789                 CAFE.MasterVehicles(i).InitialPrice(CAFE.GMYI(myPrice(k))) = CAFE.MasterVehicles(i).Price(CAFE.GMYI(myPrice(k)))
790             Next k
791
792             ' find MinYear for this vehicle--check MY col of sales and msrp
793             For k = 0 To CAFE.GMYI(MaxMY)
794                 Dim sales As Double = CAFE.MasterVehicles(i).Sales(k)
795                 Dim msrp As Double = CAFE.MasterVehicles(i).MSRP(k)
796
797                 If (Not IsNothing(sales) AndAlso sales <> 0) AndAlso_
798                    (Not IsNothing(msrp) AndAlso msrp <> 0) Then
799                     '
800                     ' found min MY col with sales and msrp values
801                     CAFE.MasterVehicles(i).MinYear = CAFE.GetMYearFromIndex(k)
802                     Exit For
803                 End If
804             Next k
805
806             'CAFE.MasterVehicles(i).SalesNotes = GetString(j, SalesNotesColNum)
807             'CAFE.MasterVehicles(i).MSRPNotes = GetString(j, MSRPNotesColNum)
808             'CAFE.MasterVehicles(i).PriceNotes = GetString(j, PriceNotesColNum)
809             CAFE.MasterVehicles(i).HybridType = GetString(j, HybridTypeColNum)
810             CAFE.MasterVehicles(i).VoltageOrPressure = GetDouble(j, VoltageOrPressureColNum)
811             CAFE.MasterVehicles(i).EnergyStorageCapacity = GetDouble(j, EnergyStorageCapacityColNum)
812             CAFE.MasterVehicles(i).BatteryType = GetString(j, BatteryTypeColNum)
813             CAFE.MasterVehicles(i).EnergyTransfer = GetString(j, EnergyTransferColNum)
814             CAFE.MasterVehicles(i).BrakingEnergyRecovery = GetDouble(j, BrakingEnergyRecoveryColNum)
815             CAFE.MasterVehicles(i).ShareMaximumPower = GetDouble(j, ShareMaximumPowerColNum)
816             'CAFE.MasterVehicles(i).HybridizationNotes = GetString(j, HybridizationNotesColNum)
817             CAFE.MasterVehicles(i).Style = GetString(j, StyleColNum)
818             CAFE.MasterVehicles(i).VehClass = GetString(j, VehClassColNum)
819             CAFE.MasterVehicles(i).VehStructure = GetString(j, VehStructureColNum)
```

```
820             CAFE.MasterVehicles(i).Drive = GetString(j, DriveColNum)
821             CAFE.MasterVehicles(i).FinalDriveRatio = GetDouble(j, FinalDriveRatioColNum)
822             CAFE.MasterVehicles(i).EngineSpeedOverVehSpeed = GetDouble(j,
        EngineSpeedOverVehSpeedColNum)
823             'CAFE.MasterVehicles(i).FrontAxleViscosity = GetString(j,
        FrontAxleViscosityColNum)
824             'CAFE.MasterVehicles(i).RearAxleViscosity = GetString(j,
        RearAxleViscosityColNum)
825             'Dim OverAllLength As Double = GetDouble(j, OverAllLengthColNum) '[KG: changed
         01/24/05 for footprint-based system]
826             'Dim OverAllWidth As Double = GetDouble(j, OverAllWidthColNum) '[KG: changed
        01/24/05 for footprint-based system]
827             Dim Wheelbase As Double = GetDouble(j, WheelbaseColNum) '[KG: changed 01/24/05
         for footprint-based system]
828             Dim TrackWidthFront As Double = GetDouble(j, TrackWidthFrontColNum) '[KG:
        changed 01/24/05 for footprint-based system]
829             Dim TrackWidthRear As Double = GetDouble(j, TrackWidthRearColNum) '[KG:
        changed 01/24/05 for footprint-based system]
830             'CAFE.MasterVehicles(i).OverAllLength = OverAllLength '[KG: changed 01/24/05
        for footprint-based system]
831             'CAFE.MasterVehicles(i).OverAllWidth = OverAllWidth '[KG: changed 01/24/05 for
         footprint-based system]
832             CAFE.MasterVehicles(i).Wheelbase = Wheelbase '[KG: changed 01/24/05 for
        footprint-based system]
833             CAFE.MasterVehicles(i).TrackWidthFront = TrackWidthFront '[KG: changed 01/24/
        05 for footprint-based system]
834             CAFE.MasterVehicles(i).TrackWidthRear = TrackWidthRear '[KG: changed 01/24/05
        for footprint-based system]
835             '[KG: changed next line 01/24/05 for footprint-based system]
836             CAFE.MasterVehicles(i).OverAllArea = Wheelbase * ((TrackWidthFront +
        TrackWidthRear) / 2) / 144 ' converts sq. in. to sq. ft.
837             CAFE.MasterVehicles(i).OverAllHeight = GetDouble(j, OverAllHeightColNum)
838             'CAFE.MasterVehicles(i).Wheelbase = GetDouble(j, WheelbaseColNum)
839             'CAFE.MasterVehicles(i).TrackWidthFront = GetDouble(j, TrackWidthFrontColNum)
840             'CAFE.MasterVehicles(i).TrackWidthRear = GetDouble(j, TrackWidthRearColNum)
841             'CAFE.MasterVehicles(i).GroundClearance = GetDouble(j, GroundClearanceColNum)
842             'CAFE.MasterVehicles(i).FrontAxleClearance = GetDouble(j,
        FrontAxleClearanceColNum)
843             'CAFE.MasterVehicles(i).RearAxleClearance = GetDouble(j,
        RearAxleClearanceColNum)
844             'CAFE.MasterVehicles(i).AngleOfApproach = GetDouble(j, AngleOfApproachColNum)
845             'CAFE.MasterVehicles(i).BreakoverAngle = GetDouble(j, BreakoverAngleColNum)
846             'CAFE.MasterVehicles(i).AngleOfDeparture = GetDouble(j,
        AngleOfDepartureColNum)
847             'CAFE.MasterVehicles(i).CenterOfGravityHeight = GetDouble(j,
        CenterOfGravityHeightColNum)
848             CAFE.MasterVehicles(i).CurbWeight = GetDouble(j, CurbWeightColNum)
849             CAFE.MasterVehicles(i).InitialCurbWeight = CAFE.MasterVehicles(i).CurbWeight
850             CAFE.MasterVehicles(i).TestWeight = GetDouble(j, TestWeightColNum)
851             CAFE.MasterVehicles(i).PAUSetting = GetDouble(j, PAUSettingColNum)
852             CAFE.MasterVehicles(i).GVWR = GetDouble(j, GVWRColNum)
853             CAFE.MasterVehicles(i).StandardTowingCapacity = GetDouble(j,
        StandardTowingCapacityColNum)
854             CAFE.MasterVehicles(i).MaxTowingCapacity = GetDouble(j,
        MaxTowingCapacityColNum)
855             CAFE.MasterVehicles(i).Payload = GetDouble(j, PayloadColNum)
856             CAFE.MasterVehicles(i).MinSeating = GetInteger(j, MinSeatingColNum)
857             CAFE.MasterVehicles(i).MaxSeating = GetInteger(j, MaxSeatingColNum)
858             'CAFE.MasterVehicles(i).FrontRowSeating = GetInteger(j, FrontRowSeatingColNum)
859             'CAFE.MasterVehicles(i).CargoVolBehindFirstRow = GetDouble(j,
        CargoVolBehindFirstRowColNum)
860             'CAFE.MasterVehicles(i).SecondRowSeating = GetInteger(j,
        SecondRowSeatingColNum)
861             'CAFE.MasterVehicles(i).SecondRowFlatCapability = GetString(j,
        SecondRowFlatCapabilityColNum)
862             'CAFE.MasterVehicles(i).CargoVolBehindSecondRow = GetDouble(j,
        CargoVolBehindSecondRowColNum)
863             'CAFE.MasterVehicles(i).ThirdRowSeating = GetInteger(j, ThirdRowSeatingColNum)
864             'CAFE.MasterVehicles(i).ThirdRowFlatCapability = GetString(j,
        ThirdRowFlatCapabilityColNum)
```

```
865             'CAFE.MasterVehicles(i).CargoVolBehindThirdRow = GetDouble(j,
    CargoVolBehindThirdRowColNum)
866             'CAFE.MasterVehicles(i).EnclosedVolume = GetDouble(j, EnclosedVolumeColNum)
867             'CAFE.MasterVehicles(i).StandardPassengerVolume = GetDouble(j,
    StandardPassengerVolumeColNum)
868             'CAFE.MasterVehicles(i).MaxPassengerVolume = GetDouble(j,
    MaxPassengerVolumeColNum)
869             'CAFE.MasterVehicles(i).CargoVolumeIndex = GetDouble(j,
    CargoVolumeIndexColNum)
870             'CAFE.MasterVehicles(i).StandardEnclosedCargoVolume = GetDouble(j,
    StandardEnclosedCargoVolumeColNum)
871             'CAFE.MasterVehicles(i).MaxEnclosedCargoVolume = GetDouble(j,
    MaxEnclosedCargoVolumeColNum)
872             'CAFE.MasterVehicles(i).OpenBoxAreaLength = GetDouble(j,
    OpenBoxAreaLengthColNum)
873             'CAFE.MasterVehicles(i).MinOpenBoxAreaWidth = GetDouble(j,
    MinOpenBoxAreaWidthColNum)
874             'CAFE.MasterVehicles(i).MaxOpenBoxAreaWidth = GetDouble(j,
    MaxOpenBoxAreaWidthColNum)
875             CAFE.MasterVehicles(i).OpenBoxArea = GetDouble(j, OpenBoxAreaColNum)
876             CAFE.MasterVehicles(i).OpenBoxHeight = GetDouble(j, OpenBoxHeightColNum)
877             CAFE.MasterVehicles(i).FuelCapacity = GetDouble(j, FuelCapacityColNum)
878             'CAFE.MasterVehicles(i).TireRollingResistance = GetDouble(j,
    TireRollingResistanceColNum)
879             'CAFE.MasterVehicles(i).FrontalArea = GetDouble(j, FrontalAreaColNum)
880             'CAFE.MasterVehicles(i).RecommendedTirePressure = GetDouble(j,
    RecommendedTirePressureColNum)
881             'CAFE.MasterVehicles(i).TireStaticLoadedRadius = GetDouble(j,
    TireStaticLoadedRadiusColNum)
882             'CAFE.MasterVehicles(i).TireTreadWidth = GetDouble(j, TireTreadWidthColNum)
883             'CAFE.MasterVehicles(i).TireDeflection = GetDouble(j, TireDeflectionColNum)
884             'CAFE.MasterVehicles(i).AerodynamicDragCoefficient = GetDouble(j,
    AerodynamicDragCoefficientColNum)
885             'CAFE.MasterVehicles(i).VehicleNotes = GetString(j, VehicleNotesColNum)
886             'CAFE.MasterVehicles(i).EPA_Class = GetString(j, EPA_ClassColNum)
887             'CAFE.MasterVehicles(i).EPA_Certification = GetInteger(j,
    EPA_CertificationColNum)
888             'CAFE.MasterVehicles(i).LEV_Class = GetString(j, LEV_ClassColNum)
889             'CAFE.MasterVehicles(i).RegulatoryNotes = GetString(j, RegulatoryNotesColNum)
890
891             CAFE.MasterVehicles(i).LTDFN1 = GetBoolean(j, LTDFN1ColNum)
892             CAFE.MasterVehicles(i).LTDFN2 = GetBoolean(j, LTDFN2ColNum)
893             CAFE.MasterVehicles(i).LTDFN3 = GetBoolean(j, LTDFN3ColNum)
894             CAFE.MasterVehicles(i).LTDFN4 = GetBoolean(j, LTDFN4ColNum)
895             CAFE.MasterVehicles(i).LTDFN5 = GetBoolean(j, LTDFN5ColNum)
896             CAFE.MasterVehicles(i).HLTINCL1 = GetBoolean(j, HLTINCL1ColNum)
897             CAFE.MasterVehicles(i).HLTINCL2 = GetBoolean(j, HLTINCL2ColNum)
898             CAFE.MasterVehicles(i).HLTINCL3 = GetBoolean(j, HLTINCL3ColNum)
899             CAFE.MasterVehicles(i).HLTINCL4 = GetBoolean(j, HLTINCL4ColNum)
900             CAFE.MasterVehicles(i).HLTINCL5 = GetBoolean(j, HLTINCL5ColNum)
901             'CAFE.MasterVehicles(i).MOBILE6Class = GetString(j, MOBILE6ClassColNum)
902             CAFE.MasterVehicles(i).SafetyClass = GetString(j, SafetyClassColNum)
903             CAFE.MasterVehicles(i).Percent2DR = GetDouble(j, Percent2DRColNum)
904             CAFE.MasterVehicles(i).MarketSegment_VOLPE = GetInteger(j,
    MarketSegment_VOLPEColNum)
905             CAFE.MasterVehicles(i).MarketSegment_AutoNews = GetInteger(j,
    MarketSegment_AutoNewsColNum)
906             If CAFE.MasterVehicles(i).MarketSegment_AutoNews = 0 Then
907                 CAFE.MasterVehicles(i).MarketSegment_AutoNews = 1
908             End If
909             'CAFE.MasterVehicles(i).MarketNest = GetInteger(j, MarketNestColNum)     '
    replaced by market segment
910
911             CAFE.MasterVehicles(i).TechOverROLL = Me.GetTechOverRide(GetString(j,
    ROLLColNum))
912             CAFE.MasterVehicles(i).TechOverLDB = Me.GetTechOverRide(GetString(j,
    LDBColNum))
913             CAFE.MasterVehicles(i).TechOverFAD = Me.GetTechOverRide(GetString(j,
    FADColNum))
914             CAFE.MasterVehicles(i).TechOverEPS = Me.GetTechOverRide(GetString(j,
```

```
EPSColNum))
915             CAFE.MasterVehicles(i).TechOverEAI = Me.GetTechOverRide(GetString(j,
EAIColNum))
916             CAFE.MasterVehicles(i).TechOverAER = Me.GetTechOverRide(GetString(j,
AERColNum))
917             CAFE.MasterVehicles(i).TechOver42V = Me.GetTechOverRide(GetString(j,
V42ColNum))
918             CAFE.MasterVehicles(i).TechOverISG = Me.GetTechOverRide(GetString(j,
ISGColNum))
919             CAFE.MasterVehicles(i).TechOverMS1 = Me.GetTechOverRide(GetString(j,
MS1ColNum))
920             CAFE.MasterVehicles(i).TechOverMS2 = Me.GetTechOverRide(GetString(j,
MS2ColNum))
921             CAFE.MasterVehicles(i).TechOverMS3 = Me.GetTechOverRide(GetString(j,
MS3ColNum))
922             CAFE.MasterVehicles(i).TechOverMSX = Me.GetTechOverRide(GetString(j,
MSxColNum))
923
924             ' Stage Analysis
925             ReDim CAFE.MasterVehicles(i).StageAnalysisFE(CAFE.GMYI(MaxStageFE, False))
926             For k = 0 To Me.StageAnalysisFEColIsMY.Length - 1
927                 CAFE.MasterVehicles(i).StageAnalysisFE(CAFE.GMYI(StageAnalysisFEColIsMY
(k))) _
928                     = GetDouble(j, Me.StageAnalysisFEColStart + k)
929             Next k
930             ReDim CAFE.MasterVehicles(i).StageAnalysisCW(CAFE.GMYI(MaxStageCW, False))
931             For k = 0 To Me.StageAnalysisCWColIsMY.Length - 1
932                 CAFE.MasterVehicles(i).StageAnalysisCW(CAFE.GMYI(StageAnalysisCWColIsMY
(k))) _
933                     = GetDouble(j, Me.StageAnalysisCWColStart + k)
934             Next k
935             ReDim CAFE.MasterVehicles(i).StageAnalysisSales(CAFE.GMYI(MaxStageSales,
False))
936             For k = 0 To Me.StageAnalysisSalesColIsMY.Length - 1
937                 CAFE.MasterVehicles(i).StageAnalysisSales(CAFE.GMYI
(StageAnalysisSalesColIsMY(k))) _
938                     = GetDouble(j, Me.StageAnalysisSalesColStart + k)
939             Next k
940         Next r
941
942         ' moved model year work to earlier in the subroutine (09-mar-04 jvs)
943
944         MyBase.ValidFile = True
945     End Sub
946
947     '''
948     ' CheckCategories:
949     ' Finds the requested category and sets return value
950     Public Sub CheckCategories(ByVal Col As Integer, ByVal ColName As String, ByRef i As
Integer, ByRef MsgString As String)
951         If Not Me.CheckRequiredColumn(Col, "    - " & ColName.Trim(), MsgString) Then
952             i += 1
953         End If
954     End Sub
955
956     '''
957     ' Validate string field
958     ' Simple check ...
959     ' If the field is blank, return FALSE and append string to FailureReason
960     Private Function ValidateField(ByVal field As String, _
961                     ByVal FieldName As String, _
962                     ByRef FailureReason As String) As Boolean
963         Dim Valid As Boolean = True
964         If field = String.Empty Then
965             Valid = False
966             FailureReason += FieldName + " is blank;"
967         End If
968         Return Valid
969     End Function
970
```

```
971     '''
972     ' Validate Double Field
973     ' Checks that the value is (a) a number, and (b) is between the specified min/max
974     Private Function ValidateField(ByVal field As Double, _
975                     ByVal Min As Double, ByVal Max As Double, _
976                     ByVal FieldName As String, _
977                     ByRef FailureReason As String) As Boolean
978         Dim valid = True
979         If field < Min Then
980             valid = False
981             FailureReason += FieldName + " is LT than min value of " + Min.ToString + ";"
982         ElseIf field > Max Then
983             valid = False
984             FailureReason += FieldName + " is GT than max value of " + Max.ToString + ";"
985         End If
986         Return valid
987     End Function
988
989     '''
990     ' Validate Integer Field
991     ' Checks that the value is (a) a number, and (b) is between the specified min/max
992     Private Function ValidateField(ByVal field As Integer, _
993                     ByVal Min As Integer, ByVal Max As Integer, _
994                     ByVal FieldName As String, _
995                     ByRef FailureReason As String) As Boolean
996         Dim valid = True
997         If field < Min Then
998             valid = False
999             FailureReason += FieldName + " is LT than min value of " + Min.ToString + ";"
1000        ElseIf field > Max Then
1001            valid = False
1002            FailureReason += FieldName + " is GT than max value of " + Max.ToString + ";"
1003        End If
1004        Return valid
1005    End Function
1006
1007    Protected Overrides Sub Finalize()
1008        MyBase.Finalize()
1009    End Sub
1010
1011    Private Function GetTechOverRide(ByVal TechOverRideField As Object) As CAFE.
        TechOverRide
1012        Dim Str As String
1013        Dim RtnCode As CAFE.TechOverRide
1014        Str = GetString(TechOverRideField).ToUpper
1015        If Str = "FORCE" Or Str = "F" Then
1016            RtnCode = CAFE.TechOverRide.Force
1017        ElseIf Str = "SKIP" Or Str = "S" Then
1018            RtnCode = CAFE.TechOverRide.Skip
1019        ElseIf Str = "USED" Or Str = "S" Then
1020            RtnCode = CAFE.TechOverRide.Used
1021        Else
1022            RtnCode = CAFE.TechOverRide.None
1023        End If
1024        Return RtnCode
1025    End Function
1026 End Class
1027
```

```
1  Public Class ScenariosClass
2
3      ' store all the scenarios found in the input files
4      Public nScenarios As Integer
5      Public Scenarios() As ScenarioClass
6
7      '
8      ' constructor
9      '
10     Public Sub New()
11         Me.nScenarios = 0
12     End Sub
13
14     '
15     ' methods
16     '
17     Public Sub Copy(ByRef Dst As ScenariosClass)
18         Copy(Dst, Me)
19     End Sub
20
21     Public Shared Sub Copy(ByRef Dst As ScenariosClass, ByRef Src As ScenariosClass)
22         Dim i, j As Integer
23
24         Dst = New ScenariosClass
25
26         ' copy the Scenarios
27         Dst.nScenarios = Src.nScenarios
28         ' resize the Scenarios array...
29         ReDim Dst.Scenarios(Dst.nScenarios - 1)
30         ' then copy it
31         For i = 0 To Dst.nScenarios - 1
32             If Not IsNothing(Src.Scenarios(i)) Then
33                 ' the i-th scenario is not blank, copy it...
34                 Src.Scenarios(i).Copy(Dst.Scenarios(i))
35             End If
36         Next i
37     End Sub
38
39     Public Sub Add_LT_RegClass_Boundaries(ByVal ScenarioIdx As Integer, ByVal index As
       Integer, ByVal val As Double)
40         Add(Scenarios(ScenarioIdx).LtRegClassBoundaries, ScenarioIdx, index, val)
41     End Sub
42
43     Public Sub Add_CAFE_RegClass_FlatSTD(ByVal ScenarioIdx As Integer, ByVal index As
       Integer, ByVal val As Double)
44         Add(Scenarios(ScenarioIdx).CafeFlatSTD, ScenarioIdx, index, val)
45     End Sub
46
47     Private Sub Add(ByRef arr()() As Double, ByVal ScenarioIdx As Integer, ByVal index As
       Integer, ByVal val As Double)
48         ' verify for valid index
49         If Not Verify_Index(ScenarioIdx, index) Then
50             Return
51         End If
52
53         ' change the size of Me.RC at index
54         If IsNothing(arr(index)) Then
55             ReDim Preserve arr(index)(0)
56         Else
57             ReDim Preserve arr(index)(arr(index).Length)
58         End If
59
60         ' add val to RC(index)
61         arr(index)(arr(index).Length - 1) = val
62     End Sub
63
64     Private Function Verify_Index(ByVal ScenarioIdx As Integer, ByVal index As Integer) As
       Boolean
65         If index > (Scenarios(ScenarioIdx).MaxYear - 2000) Or index < (Scenarios
       (ScenarioIdx).MinYear - 2000) Then
```

```
66              Throw New System.IndexOutOfRangeException("The index must be between " & _
67                                                        Scenarios(ScenarioIdx).MinYear - 2000 & " and " & _
68                                                        Scenarios(ScenarioIdx).MaxYear - 2000 & ".")
69              Return False
70          End If
71
72          Return True
73      End Function
74
75  End Class
76
77
78  Public Class ScenarioClass
79
80      ' the scenario page
81      Public ScenarioDescription As String          ' name of the scenario
82      Public nMY As Integer                         ' number of years
83      Public MY() As Double                         ' 1D array of [model] years
84      'Public Auto_CAFE_STD() As Double             ' 1D array of CAFE standards (per year)
85      Public LtDefinition() As Integer              ' 1D array of [light truck] applicability codes (per year)
86      Public HltInclusion() As Integer              ' 1D array of [heavy light truck] applicability codes (per year)
87      Public nLtRegClass() As Integer               ' number of regulatory classes
88      Public LtRegClassAttribute As String          ' attribute ["Area", "Weight", or "None"] defining light truck regulatory classes
89      Public LtRegClassBoundaries()() As Double     ' 2D array of attribute upper bounds (per class, per year)
90      Public CafeFlatSTD()() As Double              ' 2D array of CAFE flat mpg standard values (per class, per year)
91      Public TradeAllowed() As Boolean              ' 1D array of whether trading is allowed (per year)
92      Public TradeDiscount() As Double              ' 1D array of discount for trading, if applicable (per year)
93      Public LtFunctionType() As Integer            ' 1D array of functional forms (per year)
94      Public nCoefficients As Integer               ' number of coefficients
95      Public LtFunctionCoefficients(,) As Double    ' 2D array of values for coefficients (per coefficient, per year)
96      Public HLT_FlatSTD() As Double                ' 1D array of flat standards for heavy light trucks regulated as a distinct class
97      Public TransitionalFlatSTD() As Double        ' 1D array of flat standards available for light trucks during a transitional period
98      Public CafeFineRate() As Double               ' 1D array of CAFE fines values, $/mpg-vehicle (per year)
99
100     Public MinYear As Double                      ' Stores the minimum year [from the MY() array]
101     Public MaxYear As Double                      ' Stores the maximum year [from the MY() array]
102
103     ' methods
104     '
105     Public Sub Copy(ByRef Dst As ScenarioClass)
106         Copy(Dst, Me)
107     End Sub
108
109     Public Shared Sub Copy(ByRef Dst As ScenarioClass, ByRef Src As ScenarioClass)
110         Dim i, j As Integer
111
112         Dst = New ScenarioClass
113
114         ' copy the min and the max years
115         Dst.MinYear = Src.MinYear
116         Dst.MaxYear = Src.MaxYear
117
118         ' copy the scenario page
119         Dst.ScenarioDescription = Src.ScenarioDescription
120         Dst.nMY = Src.nMY
```

```
121         Dst.nLtRegClass = Src.nLtRegClass
122         Dst.nCoefficients = Src.nCoefficients
123         Dst.LtRegClassAttribute = Src.LtRegClassAttribute
124         ' redim the arrays...
125         ReDim Dst.MY(Src.nMY - 1)
126         'ReDim Dst.Auto_CAFE_STD(Src.nMY - 1)
127         ReDim Dst.LtDefinition(Src.MaxYear - 2000)
128         ReDim Dst.HltInclusion(Src.MaxYear - 2000)
129         ReDim Dst.TradeAllowed(Src.MaxYear - 2000)
130         ReDim Dst.TradeDiscount(Src.MaxYear - 2000)
131         ReDim Dst.LtFunctionType(Src.MaxYear - 2000)
132         ReDim Dst.HLT_FlatSTD(Src.MaxYear - 2000)
133         ReDim Dst.TransitionalFlatSTD(Src.MaxYear - 2000)
134         ReDim Dst.CafeFineRate(Src.MaxYear - 2000)
135         ReDim Dst.LtRegClassBoundaries(Src.LtRegClassBoundaries.Length - 1)
136         ReDim Dst.CafeFlatSTD(Src.CafeFlatSTD.Length - 1)
137         For i = 0 To Src.LtRegClassBoundaries.Length - 1          ' outer, my, loop
138             If Not IsNothing(Src.LtRegClassBoundaries(i)) Then
139                 ReDim Dst.LtRegClassBoundaries(i)(Src.LtRegClassBoundaries(i).Length - 1)
140                 For j = 0 To Src.LtRegClassBoundaries(i).Length - 1    ' inner, my(i), loop
141                     Dst.LtRegClassBoundaries(i)(j) = Src.LtRegClassBoundaries(i)(j)
142                 Next j
143             End If
144         Next i
145         For i = 0 To Src.CafeFlatSTD.Length - 1
146             If Not IsNothing(Src.CafeFlatSTD(i)) Then
147                 ReDim Dst.CafeFlatSTD(i)(Src.CafeFlatSTD(i).Length - 1)
148                 For j = 0 To Src.CafeFlatSTD(i).Length - 1
149                     Dst.CafeFlatSTD(i)(j) = Src.CafeFlatSTD(i)(j)
150                 Next j
151             End If
152         Next i
153         ReDim Dst.LtFunctionCoefficients(Src.MaxYear - 2000, Src.nCoefficients - 1)
154         ' then copy them
155         For j = 0 To Src.nMY - 1
156             Dim m As Integer = Src.MY(j) - 2000
157             Dst.MY(j) = Src.MY(j)
158             'Dst.Auto_CAFE_STD(j) = Src.Auto_CAFE_STD(j)
159             Dst.LtDefinition(m) = Src.LtDefinition(m)
160             Dst.HltInclusion(m) = Src.HltInclusion(m)
161             Dst.TradeAllowed(m) = Src.TradeAllowed(m)
162             Dst.TradeDiscount(m) = Src.TradeDiscount(m)
163             Dst.LtFunctionType(m) = Src.LtFunctionType(m)
164             Dst.HLT_FlatSTD(m) = Src.HLT_FlatSTD(m)
165             Dst.TransitionalFlatSTD(m) = Src.TransitionalFlatSTD(m)
166             Dst.CafeFineRate(m) = Src.CafeFineRate(m)
167
168             For i = 0 To Src.nCoefficients - 1
169                 Dst.LtFunctionCoefficients(m, i) = Src.LtFunctionCoefficients(m, i)
170             Next i
171         Next j
172     End Sub
173
174 End Class
175
```

```
Public Class m_ShowFilesForm
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub
    Public Sub New(ByVal M As String, _
                   ByVal V As String, ByVal E As String, ByVal T As String, _
                   ByVal Tn As String, ByVal P As String, ByVal B As String)
        '
        Me.New()
        Me.FillForm(M, V, E, T, Tn, P, B)
    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents Label4 As System.Windows.Forms.Label
    Friend WithEvents Label5 As System.Windows.Forms.Label
    Friend WithEvents m_DisplayVehiclesFile As System.Windows.Forms.Label
    Friend WithEvents m_DisplayEnginesFile As System.Windows.Forms.Label
    Friend WithEvents m_DisplayTransmissionsFile As System.Windows.Forms.Label
    Friend WithEvents m_DisplayTechnologyFile As System.Windows.Forms.Label
    Friend WithEvents m_DisplayParametersFile As System.Windows.Forms.Label
    Friend WithEvents m_CloseButton As System.Windows.Forms.Button
    Friend WithEvents Label6 As System.Windows.Forms.Label
    Friend WithEvents m_DisplayMarketDataFile As System.Windows.Forms,Label
    Friend WithEvents Label0 As System.Windows.Forms.Label
    Friend WithEvents m_DisplayScenariosFile As System.Windows.Forms.Label
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label
        Me.Label2 = New System.Windows.Forms.Label
        Me.Label3 = New System.Windows.Forms.Label
        Me.Label4 = New System.Windows.Forms.Label
        Me.Label5 = New System.Windows.Forms.Label
        Me.m_DisplayVehiclesFile = New System.Windows.Forms.Label
        Me.m_DisplayEnginesFile = New System.Windows.Forms.Label
        Me.m_DisplayTransmissionsFile = New System.Windows.Forms.Label
        Me.m_DisplayTechnologyFile = New System.Windows.Forms.Label
        Me.m_DisplayParametersFile = New System.Windows.Forms.Label
        Me.m_CloseButton = New System.Windows.Forms.Button
        Me.m_DisplayScenariosFile = New System.Windows.Forms.Label
        Me.Label6 = New System.Windows.Forms.Label
        Me.Label0 = New System.Windows.Forms.Label
        Me.m_DisplayMarketDataFile = New System.Windows.Forms.Label
        Me.SuspendLayout()
```

```
71          '
72          'Label1
73          '
74          Me.Label1.Location = New System.Drawing.Point(16, 40)
75          Me.Label1.Name = "Label1"
76          Me.Label1.Size = New System.Drawing.Size(200, 16)
77          Me.Label1.TabIndex = 0
78          Me.Label1.Text = "Vehicle Models Dataset File:"
79          Me.Label1.TextAlign = System.Drawing.ContentAlignment.MiddleRight
80          '
81          'Label2
82          '
83          Me.Label2.Location = New System.Drawing.Point(16, 64)
84          Me.Label2.Name = "Label2"
85          Me.Label2.Size = New System.Drawing.Size(200, 16)
86          Me.Label2.TabIndex = 1
87          Me.Label2.Text = "Vehicle Engines Dataset File:"
88          Me.Label2.TextAlign = System.Drawing.ContentAlignment.MiddleRight
89          '
90          'Label3
91          '
92          Me.Label3.Location = New System.Drawing.Point(16, 88)
93          Me.Label3.Name = "Label3"
94          Me.Label3.Size = New System.Drawing.Size(200, 16)
95          Me.Label3.TabIndex = 2
96          Me.Label3.Text = "Vehicle Transmissions Dataset File:"
97          Me.Label3.TextAlign = System.Drawing.ContentAlignment.MiddleRight
98          '
99          'Label4
100         '
101         Me.Label4.Location = New System.Drawing.Point(16, 128)
102         Me.Label4.Name = "Label4"
103         Me.Label4.Size = New System.Drawing.Size(200, 16)
104         Me.Label4.TabIndex = 3
105         Me.Label4.Text = "Technology File:"
106         Me.Label4.TextAlign = System.Drawing.ContentAlignment.MiddleRight
107         '
108         'Label5
109         '
110         Me.Label5.Location = New System.Drawing.Point(16, 152)
111         Me.Label5.Name = "Label5"
112         Me.Label5.Size = New System.Drawing.Size(200, 16)
113         Me.Label5.TabIndex = 4
114         Me.Label5.Text = "Parameters File:"
115         Me.Label5.TextAlign = System.Drawing.ContentAlignment.MiddleRight
116         '
117         'm_DisplayVehiclesFile
118         '
119         Me.m_DisplayVehiclesFile.AutoSize = True
120         Me.m_DisplayVehiclesFile.Location = New System.Drawing.Point(240, 40)
121         Me.m_DisplayVehiclesFile.Name = "m_DisplayVehiclesFile"
122         Me.m_DisplayVehiclesFile.Size = New System.Drawing.Size(97, 16)
123         Me.m_DisplayVehiclesFile.TabIndex = 6
124         Me.m_DisplayVehiclesFile.Text = "{No File Specified}"
125         Me.m_DisplayVehiclesFile.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
126         '
127         'm_DisplayEnginesFile
128         '
129         Me.m_DisplayEnginesFile.AutoSize = True
130         Me.m_DisplayEnginesFile.Location = New System.Drawing.Point(240, 64)
131         Me.m_DisplayEnginesFile.Name = "m_DisplayEnginesFile"
132         Me.m_DisplayEnginesFile.Size = New System.Drawing.Size(97, 16)
133         Me.m_DisplayEnginesFile.TabIndex = 7
134         Me.m_DisplayEnginesFile.Text = "{No File Specified}"
135         Me.m_DisplayEnginesFile.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
136         '
137         'm_DisplayTransmissionsFile
138         '
139         Me.m_DisplayTransmissionsFile.AutoSize = True
140         Me.m_DisplayTransmissionsFile.Location = New System.Drawing.Point(240, 88)
```

```
141         Me.m_DisplayTransmissionsFile.Name = "m_DisplayTransmissionsFile"
142         Me.m_DisplayTransmissionsFile.Size = New System.Drawing.Size(97, 16)
143         Me.m_DisplayTransmissionsFile.TabIndex = 8
144         Me.m_DisplayTransmissionsFile.Text = "{No File Specified}"
145         Me.m_DisplayTransmissionsFile.TextAlign = System.Drawing.ContentAlignment.
        MiddleLeft
146         '
147         'm_DisplayTechnologyFile
148         '
149         Me.m_DisplayTechnologyFile.AutoSize = True
150         Me.m_DisplayTechnologyFile.Location = New System.Drawing.Point(240, 128)
151         Me.m_DisplayTechnologyFile.Name = "m_DisplayTechnologyFile"
152         Me.m_DisplayTechnologyFile.Size = New System.Drawing.Size(97, 16)
153         Me.m_DisplayTechnologyFile.TabIndex = 9
154         Me.m_DisplayTechnologyFile.Text = "{No File Specified}"
155         Me.m_DisplayTechnologyFile.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
156         '
157         'm_DisplayParametersFile
158         '
159         Me.m_DisplayParametersFile.AutoSize = True
160         Me.m_DisplayParametersFile.Location = New System.Drawing.Point(240, 152)
161         Me.m_DisplayParametersFile.Name = "m_DisplayParametersFile"
162         Me.m_DisplayParametersFile.Size = New System.Drawing.Size(97, 16)
163         Me.m_DisplayParametersFile.TabIndex = 10
164         Me.m_DisplayParametersFile.Text = "{No File Specified}"
165         Me.m_DisplayParametersFile.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
166         '
167         'm_CloseButton
168         '
169         Me.m_CloseButton.Location = New System.Drawing.Point(256, 208)
170         Me.m_CloseButton.Name = "m_CloseButton"
171         Me.m_CloseButton.Size = New System.Drawing.Size(96, 32)
172         Me.m_CloseButton.TabIndex = 12
173         Me.m_CloseButton.Text = "&Close"
174         '
175         'm_DisplayScenariosFile
176         '
177         Me.m_DisplayScenariosFile.AutoSize = True
178         Me.m_DisplayScenariosFile.Location = New System.Drawing.Point(240, 176)
179         Me.m_DisplayScenariosFile.Name = "m_DisplayScenariosFile"
180         Me.m_DisplayScenariosFile.Size = New System.Drawing.Size(97, 16)
181         Me.m_DisplayScenariosFile.TabIndex = 11
182         Me.m_DisplayScenariosFile.Text = "{No File Specified}"
183         Me.m_DisplayScenariosFile.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
184         '
185         'Label6
186         '
187         Me.Label6.Location = New System.Drawing.Point(16, 176)
188         Me.Label6.Name = "Label6"
189         Me.Label6.Size = New System.Drawing.Size(200, 16)
190         Me.Label6.TabIndex = 5
191         Me.Label6.Text = "Scenarios File:"
192         Me.Label6.TextAlign = System.Drawing.ContentAlignment.MiddleRight
193         '
194         'Label0
195         '
196         Me.Label0.Location = New System.Drawing.Point(16, 16)
197         Me.Label0.Name = "Label0"
198         Me.Label0.Size = New System.Drawing.Size(200, 16)
199         Me.Label0.TabIndex = 13
200         Me.Label0.Text = "Market Data Dataset File:"
201         Me.Label0.TextAlign = System.Drawing.ContentAlignment.MiddleRight
202         '
203         'm_DisplayMarketDataFile
204         '
205         Me.m_DisplayMarketDataFile.AutoSize = True
206         Me.m_DisplayMarketDataFile.Location = New System.Drawing.Point(240, 16)
207         Me.m_DisplayMarketDataFile.Name = "m_DisplayMarketDataFile"
208         Me.m_DisplayMarketDataFile.Size = New System.Drawing.Size(97, 16)
209         Me.m_DisplayMarketDataFile.TabIndex = 14
```

```
210         Me.m_DisplayMarketDataFile.Text = "{No File Specified}"
211         '
212         'm_ShowFilesForm
213         '
214         Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
215         Me.ClientSize = New System.Drawing.Size(368, 253)
216         Me.ControlBox = False
217         Me.Controls.Add(Me.m_DisplayMarketDataFile)
218         Me.Controls.Add(Me.Label10)
219         Me.Controls.Add(Me.m_DisplayScenariosFile)
220         Me.Controls.Add(Me.Label6)
221         Me.Controls.Add(Me.m_CloseButton)
222         Me.Controls.Add(Me.m_DisplayParametersFile)
223         Me.Controls.Add(Me.m_DisplayTechnologyFile)
224         Me.Controls.Add(Me.m_DisplayTransmissionsFile)
225         Me.Controls.Add(Me.m_DisplayEnginesFile)
226         Me.Controls.Add(Me.m_DisplayVehiclesFile)
227         Me.Controls.Add(Me.Label5)
228         Me.Controls.Add(Me.Label4)
229         Me.Controls.Add(Me.Label3)
230         Me.Controls.Add(Me.Label2)
231         Me.Controls.Add(Me.Label1)
232         Me.Name = "m_ShowFilesForm"
233         Me.Text = "Loaded Data Files"
234         Me.ResumeLayout(False)
235
236     End Sub
237
238 #End Region
239
240     Private MaxHeight As Integer
241     Private MaxWidth As Integer
242
243     Public Sub FillForm(ByVal M As String,
244                         ByVal V As String, ByVal E As String, ByVal T As String, _
245                         ByVal Tn As String, ByVal P As String, ByVal S As String)
246         '
247         ' The entry args are the text to display
248         Me.MaxHeight = 16
249         Me.MaxWidth = 0
250
251         With CAFE.OtherSettings
252             Me.SetFileText(M, Me.Label10, Me.m_DisplayMarketDataFile, .
    SingleFileDatasetInput)    ' Market Data file name
253             Me.SetFileText(V, Me.Label1, Me.m_DisplayVehiclesFile, Not .
    SingleFileDatasetInput) ' Models file name
254             Me.SetFileText(E, Me.Label2, Me.m_DisplayEnginesFile, Not .
    SingleFileDatasetInput)  ' Engines file name
255             Me.SetFileText(T, Me.Label3, Me.m_DisplayTransmissionsFile, Not .
    SingleFileDatasetInput)    ' Transmissions file name
256             Me.SetFileText(Tn, Me.Label4, Me.m_DisplayTechnologyFile, True)
          ' Technologies file name
257             Me.SetFileText(P, Me.Label5, Me.m_DisplayParametersFile, True)
          ' Parameters file name
258             Me.SetFileText(S, Me.Label6, Me.m_DisplayScenariosFile, True)
          ' Scenarios file name
259         End With
260
261         ' adjust the form geometry...
262         Me.MaxWidth = Me.m_DisplayParametersFile.Left + MaxWidth + 40
263         If Me.MaxWidth < 376 Then
264             Me.MaxWidth = 376
265         End If
266
267         Me.Width = Me.MaxWidth
268         Me.Height = Me.MaxHeight + 8 + Me.m_CloseButton.Height + 40
269
270         ' place close button 24 pixels from the right and 40 pixels from the bottom...
271         Me.m_CloseButton.Left = Me.Width - Me.m_CloseButton.Width - 24
272         Me.m_CloseButton.Top = Me.Height - Me.m_CloseButton.Height - 40
```

```
273     End Sub
274
275     '''
276     ' Sets the text of the specified Control to the specified FileName.  The specified
        Label and
277     ' Control are hidden if the Visible parameter is false.  If Visible is false, the
        MaxHeight of
278     ' the form is not calculated.  The MaxWidth of the form is set if the resulting width
        of the
279     ' specified control is greater than the width of any previous control.
280     '
281     Private Sub SetFileText(ByVal FileName As String, _
282                             ByRef Label As System.Windows.Forms.Label, _
283                             ByRef Control As System.Windows.Forms.Label, _
284                             ByVal Visible As Boolean)
285         '
286         ' set text
287         If FileName <> "" Then
288             Control.Text = FileName
289         Else
290             Control.Text = "(No file specified)"
291         End If
292
293         ' set visible status
294         Label.Visible = Visible
295         Control.Visible = Visible
296
297         ' set y location
298         Label.Top = Me.MaxHeight
299         Control.Top = Me.MaxHeight
300
301         ' adjust max height
302         If Visible Then
303             Me.MaxHeight += 24
304         End If
305
306         ' adjust max width
307         If Control.Width > MaxWidth Then
308             MaxWidth = Control.Width
309         End If
310     End Sub
311
312     Private Sub m_CloseButton_Click(ByVal sender As Object, ByVal e As System.EventArgs)
        Handles m_CloseButton.Click
313         Me.Close()
314     End Sub
315
316     Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles Label1.Click
317
318     End Sub
319 End Class
320
```

```
Public Class STD
    ' This class contains the following functions related to the CAFE program:
    ' - GetTarget:  report the fuel economy "target" for a specific vehicle
    ' - GetStandard:  report the CAFE standard applicable to a specific regulatory class
    ' - GetRegClass:  report the regulatory class for a specific vehicle

    'Public Function GetTarget(ByVal Scenario As Integer, _
    '                          ByVal MY As Integer, _
    '                          ByVal iMfg As Integer, _
    '                          ByVal iVeh As Integer) As Double
    '    '
    '    ' Find and return the fuel economy target (mpg) for this vehicle.
    '    '
    '    ' NOTE:  This function refers to the "current" vehicle attributes (e.g., weight),
    '    '        including the vehicle's regulatory class, which is assumed to be correct.
    '    '        Therefore, finding the fuel economy target that would apply
    '    '        to a pending "test" vehicle requires that the client temporarily
    '    '        substitute the relevant attributes for those in the "current"
    '    '        vehicles data structure, and then reevaluate and reassign the reg. class
    '    '        before calculating the target.  If the client uses any such temporary
    '    '        substitutions, the regulatory class should be reevaluated (and reassigned)
    '    '        immediately after the temporary substitution is reversed.
    '    '
    '    ' Initialization
    '    ' Find the relevant attributes of this vehicle.
    '    Dim Area As Double = CAFE.Vehicles(iVeh).OverAllArea ' TODO:  add to vehicles
    '    Dim Weight As Double = CAFE.Vehicles(iVeh).CurbWeight
    '    Return GetTarget(Scenario, MY, iMfg, iVeh, Area, Weight)
    'End Function    ' GetTarget (Scenario, MY, iMfg, iVeh)

    Public Function GetTarget(ByVal Scenario As Integer, _
                              ByVal MY As Integer, _
                              ByVal iMfg As Integer, _
                              ByVal iVeh As Integer, _
                              Optional ByVal Area As Double = Double.NaN, _
                              Optional ByVal Weight As Double = Double.NaN)
        '
        ' Find and return the fuel economy target (mpg) for this vehicle.
        '
        ' NOTE:  This function refers to the "current" vehicle attributes (e.g., weight),
        '        including the vehicle's regulatory class, which is assumed to be correct.
        '        Therefore, finding the fuel economy target that would apply
        '        to a pending "test" vehicle requires that the client temporarily
        '        substitute the relevant attributes for those in the "current"
        '        vehicles data structure, and then reevaluate and reassign the reg. class
        '        before calculating the target.  If the client uses any such temporary
        '        substitutions, the regulatory class should be reevaluated (and reassigned)
        '        immediately after the temporary substitution is reversed.
        '
        ' Initialization
        ' If Area and/or Weight aren't supplied, use current veh's settings
        Dim vehArea As Double
        Dim vehWeight As Double
        If Double.IsNaN(Area) Then
            vehArea = CAFE.Vehicles(iVeh).OverAllArea ' TODO:  add to vehicles
        Else
            vehArea = Area
        End If
        If Double.IsNaN(Weight) Then
            vehWeight = CAFE.Vehicles(iVeh).CurbWeight
        Else
            vehWeight = Weight
        End If

        Dim Target As Double = Double.NaN ' If NaN is returned, catch externally as an error.
        ' Get functional form and coefficients applicable to this vehicle in this MY.
        ' NOTE:  FuncForm should be externally set to 0 (flat standard) for domestic
        '        automobiles (class 1) and imported automobiles (class 2).  If this
```

```
69          '       scenario involves two or more subclasses (3-10) of light trucks, FuncForm
70          '       should also be set to 0 for all light trucks.  If light trucks are
71          '       covered as a single class (class 3) in this scenario, other functional
72          '       forms (1-8) can be specified.  Future versions could provide for
    functional
73          '       forms at the subclass level.
74          ' TODO: resolve links to parameters for the CAFE program
75          Dim RegClass As Integer = CAFE.Vehicles(iVeh).RegClass ' TODO:  add to vehicles
76          ' Dim nRegClass As Integer = CAFE.Parameters.Scenarios(Scenario).nLtRegClass(CAFE.
    GMYI(MY)) + 3 ' total number reg. classes
77          Dim FlatStd() As Double = CAFE.Scenarios.Scenarios(Scenario).CafeFlatSTD(CAFE.GMYI
    (MY)) ' flat standards (by class)
78          Dim HLT_FlatStd As Double = CAFE.Scenarios.Scenarios(Scenario).HLT_FlatSTD(CAFE.
    GMYI(MY)) ' flat standards for HLTs
79          Dim LT_FunctionType As Integer = CAFE.Scenarios.Scenarios(Scenario).LtFunctionType
    (CAFE.GMYI(MY))
80          Dim i As Integer                    ' counter variable
81          Dim uBound As Integer = CAFE.Scenarios.Scenarios(Scenario).LtFunctionCoefficients.
    GetUpperBound(1)
82          Dim Coeff(uBound) As Double      ' by coeff. index
83          For i = 0 To uBound - 1
84              Coeff(i) = CAFE.Scenarios.Scenarios(Scenario).LtFunctionCoefficients(CAFE.GMYI
    (MY), i)
85          Next i
86          Dim Fn1iVeh As Integer = 0 ' vehicle index used under fn. 1
87          Dim Fn1Sales As Double = 0 ' accumulator for MY2002 sales under fn. 1
88          Dim Fn1Denominator As Double = 0 ' accumulator for MY2002 sales x weight under fn.
    1
89          '
90          ' Processing
91          If RegClass = 1 OrElse RegClass = 2 Then
92              ' This is a passenger automobile, so it is subject to a flat standard (CarStd).
93              Target = FlatStd(RegClass)
94          ElseIf RegClass = 11 Then
95              ' This is a regulated heavy light truck, and such vehicles are regulated as
96              ' a distinct regulatory class.
97              Target = HLT_FlatStd
98          ElseIf RegClass >= 3 AndAlso LT_FunctionType = 0 Then
99              ' This is a light truck subject to a flat standard.
100             Target = FlatStd(RegClass)
101         ElseIf IsNothing(CAFE.Mfgs(iMfg).RC(RegClass)) = False Then
102             ' This is a light truck subject to a functional standard.
103             Select Case LT_FunctionType
104                 Case 1 ' fixed attribute standard based on MY2002 curb weights
105                     ' TODO:  This type of standard needs to be evaluated once at the front
106                     ' end, and subsequently treated as a manufacturer-specific flat
    standard.
107                     ' This is because the curb weight is allowed to change from model year
108                     ' to model year, but the calculation needs to be based on MY2002 curb
109                     ' weights (and sales).  Until then, return NaN
110                     ' It's also inefficient to keep repeating this calculation.
111                     'For i = 0 To CAFE.Mfgs(iMfg).TruckIndexes.Length
112                     '    Fn1iVeh = CAFE.Mfgs(iMfg).TruckIndexes(i)
113                     '    Fn1Sales += CAFE.Vehicles(Fn1iVeh).Sales(2002)
114                     '    Fn1Denominator += CAFE.Vehicles(Fn1iVeh).Sales(2002) * _
115                     '                      CAFE.Vehicles(Fn1iVeh).CurbWeight
116                     'Next
117                     'Target = Coeff(0) * Fn1Sales / Fn1Denominator
118                 Case 2 ' logistic weight-based standard
119                     Target = 1 / ((1 / Coeff(0)) + (1 / Coeff(1) - (1 / Coeff(0))) * _
120                     (Math.Exp((vehWeight - Coeff(3)) / Coeff(2)) / _
121                     (1 + Math.Exp((vehWeight - Coeff(3)) / Coeff(2)))))
122                 Case 3 ' exponential weight-based standard
123                     Target = 1 / ((1 / Coeff(0)) - (1 / Coeff(1)) * _
124                     Math.Exp(1 - vehWeight / Coeff(2)))
125                 Case 4 ' logistic area-based standard
126                     Target = 1 / ((1 / Coeff(0)) + (1 / Coeff(1) - (1 / Coeff(0))) * _
127                     (Math.Exp((vehArea - Coeff(3)) / Coeff(2)) / _
128                     (1 + Math.Exp((vehArea - Coeff(3)) / Coeff(2)))))
129                 Case 5 ' exponential area-based standard
```

```
130                     Target = 1 / ((1 / Coeff(0)) - (1 / Coeff(1)) * _
131                     Math.Exp(1 - vehArea / Coeff(2)))
132                 Case 6 ' logistic weight- and area- based standard
133                     Target = 1 / ((1 / Coeff(0)) + (1 / Coeff(1) - (1 / Coeff(0))) * _
134                     (Math.Exp((vehArea - Coeff(3)) / Coeff(2)) / _
135                     (1 + Math.Exp((vehArea - Coeff(3)) / Coeff(2)))) * _
136                     (Math.Exp((vehArea - Coeff(5)) / Coeff(4)) / _
137                     (1 + Math.Exp((vehArea - Coeff(5)) / Coeff(4)))))
138                 Case 7 ' exponential weight- and area-based standard
139                     Target = 1 / ((1 / Coeff(0)) - (1 / Coeff(1)) * _
140                     Math.Exp(1 - vehWeight / Coeff(2) - vehArea / Coeff(3)))
141                 Case 8 ' weight-based function with extra credit for weight efficiency
142                     Target = (1 / Coeff(0)) * (vehWeight / Coeff(1) + Coeff(2)) * _
143                     (Coeff(3) - vehWeight / (Coeff(4) * vehArea - Coeff(5)))
144                 Case 9 ' bin-based (i.e., stepwise) function derived from regulatory class ↙
    targets
145                     Dim TruckClassType As String = CAFE.Scenarios.Scenarios(Scenario). ↙
    LtRegClassAttribute
146                     Dim TruckClassMax() As Double = CAFE.Scenarios.Scenarios(Scenario). ↙
    LtRegClassBoundaries(CAFE.GMYI(MY)) ' by model year
147                     Dim AssignUsing As Double ' temporary variable for value of relevant ↙
    attribute
148                     If TruckClassType = "Area" Then AssignUsing = vehArea Else AssignUsing ↙
    = vehWeight
149                     If AssignUsing <= TruckClassMax(3) Then
150                         Target = FlatStd(3) '1st truck bin
151                     ElseIf AssignUsing <= TruckClassMax(4) Then
152                         Target = FlatStd(4) '2nd of multiple light bins
153                     ElseIf AssignUsing <= TruckClassMax(5) Then
154                         Target = FlatStd(5) '3nd of multiple light truck bins
155                     ElseIf AssignUsing <= TruckClassMax(6) Then
156                         Target = FlatStd(6) '4th of multiple light truck bins
157                     ElseIf AssignUsing <= TruckClassMax(7) Then
158                         Target = FlatStd(7) '5th of multiple light truck bins
159                     ElseIf AssignUsing <= TruckClassMax(8) Then
160                         Target = FlatStd(8) '6th of multiple light truck bins
161                     ElseIf AssignUsing <= TruckClassMax(9) Then
162                         Target = FlatStd(9) '7th of multiple light truck bins
163                     Else
164                         Target = FlatStd(10) '8th of 8 light truck classes
165                     End If ' End assignment of bin-based target
166                 Case Else
167                     ' return NaN
168             End Select
169         Else
170             ' One of the following problems applies:
171             ' - This vehicle is unregulated (i.e., RegClass=0).
172             ' - Light trucks have been subdivided (i.e., RegClass>3) and a functional ↙
    standard has
173             '   been requested, but we don't currently handle functional standards for ↙
    multiple light truck classes.
174             ' In either case, leave Target=NaN, as initialized.
175         End If
176         '
177         ' Termination
178         Return Target
179         '
180     End Function ' GetTarget (Scenario, MY, iMfg, iVeh, Area, Weight)
181
182     Public Function GetStandard(ByVal Scenario As Integer, _
183                                 ByVal MY As Integer, _
184                                 ByVal iMfg As Integer, _
185                                 ByVal RegClass As Integer, _
186                                 ByVal nTestVeh As Integer, _
187                                 ByVal TestVehIdx() As Integer, _
188                                 ByVal TestVehArea() As Double, _
189                                 ByVal TestVehWeight() As Double) As Double
190         ' Calculate and return the CAFE standard applicable to the specified manufacturer
191         ' in this model year for the specified regulatory class under the specified
192         ' CAFE program scenario.
```

```
193         '
194         ' Currently supported regulatory classes are:
195         '    1:  domestic passenger automobiles
196         '    2:  imported passenger automobiles
197         ' 3-10:  light trucks
198         '   11:  heavy light trucks regulated as a distinct class
199         '
200         ' NOTE:  Unregulated vehicles are assigned regulatory class 0.  GetStandard returns
201         '        NaN if called for RegClass = 0.
202         '
203         ' NOTE:  This function uses the current vehicle regulatory class assignments,
204         '        which are assumed to be correct.
205         '        Therefore, finding the fuel economy standard that would apply
206         '        to a given regulatory class under a pending application of technologies
207         '        requires that the client temporarily substitute the relevant attributes for
208         '        those in the "current" vehicles data structure, and then reevaluate and
209         '        reassigne regulatory classes before calculating the target.  If the client
210         '        uses(any) such temporary substitutions, regulatory classes should be
211         '        reevaluated and reassigned immediately after the temporary substitution
212         '        is reversed.
213         '
214         ' Initialization
215         Dim CAFE_Standard As Double = Double.NaN ' trap NaN externally as an error
216         Dim Sales As Integer = 0 ' accumulator for sales in this regulatory class
217         Dim StdSum As Double = 0 ' accumulator for sales/target in this regulatory class
218         Dim i As Integer = 0 ' counter
219         Dim iVeh As Integer = 0 ' vehicle index
220         Dim Target As Double = Double.NaN
221         ' TODO:  resolve links to parameters for the CAFE program
222         Dim LT_FunctionType As Integer = CAFE.Scenarios.Scenarios(Scenario).LtFunctionType(CAFE.GMYI(MY))
223         ' Dim nRegClass As Integer = CAFE.Parameters.Scenarios(Scenario).nLtRegClass(CAFE.GMYI(MY)) + 3 ' total number reg. classes
224         Dim FlatStd() As Double = CAFE.Scenarios.Scenarios(Scenario).CafeFlatSTD(CAFE.GMYI(MY)) ' flat standards (by class)
225         Dim HLT_FlatStd As Double = CAFE.Scenarios.Scenarios(Scenario).HLT_FlatSTD(CAFE.GMYI(MY)) ' flat standards for HLTs
226         Dim TransitionalFlatSTD As Double = CAFE.Scenarios.Scenarios(Scenario).TransitionalFlatSTD(CAFE.GMYI(MY)) ' transitional LT standard
227         '
228         ' Processing
229         If RegClass = 0 Then
230             ' These are unregulated vehicles
231             CAFE_Standard = 0
232         ElseIf RegClass = 1 OrElse RegClass = 2 Then
233             ' These are passenger automobiles subject to a flat standard (CarStd).
234             CAFE_Standard = FlatStd(RegClass)
235         ElseIf RegClass = 11 Then
236             ' This is a regulated heavy light truck, and such vehicles are regulated as
237             ' a distinct regulatory class.
238             CAFE_Standard = HLT_FlatStd
239         ElseIf RegClass >= 3 AndAlso LT_FunctionType = 0 Then
240             ' Light trucks are covered by flat standards.
241             CAFE_Standard = FlatStd(RegClass)
242         ElseIf RegClass >= 3 AndAlso LT_FunctionType = 1 Then
243             ' Light trucks are a single class, but manufacturer-specific "fixed attribute"
244             ' standards apply.  Currently, these standards are defined as the product of a constant
245             ' and the manufacturer's sales-weighted average curb weight in MY 2002 or, for new entrants,
246             ' the industrywide sales-weighted average curb weight in MY 2002.
247             CAFE_Standard = CAFE.Scenarios.Scenarios(Scenario).LtFunctionCoefficients(CAFE.GMYI(MY), 0) / _
248                             CAFE.Parameters.MY02LTWeight(iMfg)
249         ElseIf IsNothing(CAFE.Mfgs(iMfg).RC(RegClass)) = False Then
250             ' Light trucks are a single class, but a functional standard applies,
251             ' so the vehicle-specific "targets" must be evaluated and combined
252             ' using a sales-weighted harmonic average.
253             ' Go through vehicles and assign sales and sales/target values to running
```

```
    totals.
254             For i = 0 To CAFE.Mfgs(iMfg).RC(RegClass).Length - 1
255                 ' Get vehicle index
256                 iVeh = CAFE.Mfgs(iMfg).RC(RegClass)(i)
257                 ' Do a quick check to see if we have test vehicles
258                 If nTestVeh = 0 Then
259                     ' Get fuel economy target for current vehicle.
260                     Target = GetTarget(Scenario, MY, iMfg, iVeh)
261                 Else
262                     ' Need to check if this vehicle is on the test vehicle list
263                     Dim tidx As Integer = Array.IndexOf(TestVehIdx, iVeh)
264                     If Array.IndexOf(TestVehIdx, iVeh) >= 0 Then
265                         ' Found this vehicle on the test list
266                         Target = GetTarget(Scenario, MY, iMfg, TestVehIdx(tidx),
    TestVehArea(tidx), TestVehWeight(tidx))
267                     Else
268                         ' Vehicle not on the list.
269                         Target = GetTarget(Scenario, MY, iMfg, iVeh)
270                     End If
271                 End If
272                 ' TODO:  Add error handling if Target = NaN at this point.
273                 ' Add sales to cumulative for appropriate class.
274                 Sales += CAFE.Vehicles(iVeh).Sales(CAFE.GMYI(MY))
275                 ' Add sales/target to cumulative for appropriate class.
276                 StdSum += CAFE.Vehicles(iVeh).Sales(CAFE.GMYI(MY)) / Target
277             Next i
278             ' Calculate the standard.
279             CAFE_Standard = Sales / StdSum
280         Else
281             ' One of the following problems applies:
282             ' - This class is unregulated (i.e., RegClass=0).
283             ' - Light trucks have been subdivided (i.e., RegClass>3) and a functional
    standard has
284             '   been requested, but we don't currently handle functional standards for
    multiple light truck classes.
285             ' In either case, leave CAFE_Standard=NaN, as initialized.
286         End If
287         '
288         ' Round CAFE standard to nearest 0.1 mpg.
289         CAFE_Standard = Math.Round(CAFE_Standard, 1)
290
291         ' If available, offer transitional standard.
292         TransitionalFlatSTD = Math.Round(TransitionalFlatSTD, 1)
293         If RegClass = 3 AndAlso TransitionalFlatSTD > 0 Then
294             If TransitionalFlatSTD < CAFE_Standard Then
295                 CAFE_Standard = TransitionalFlatSTD
296             End If
297         End If
298
299
300         '
301         ' Termination
302         Return CAFE_Standard
303         '
304     End Function ' GetStandard
305
306     Public Function GetStandard(ByVal Scenario As Integer, _
307                                 ByVal MY As Integer, _
308                                 ByVal iMfg As Integer, _
309                                 ByVal RegClass As Integer) As Double
310         '
311         Return GetStandard(Scenario, MY, iMfg, RegClass, 0, Nothing, Nothing, Nothing)
312     End Function ' GetStandard (w/test vehicles)
313
314     Public Function GetRegClass(ByVal Scenario As Integer, _
315                                 ByVal MY As Integer, _
316                                 ByVal iVeh As Integer, _
317                                 Optional ByRef TestCurbWeight As Double = Double.NaN) As
    Integer
318         '
```

```
319         'This fcn. assigns a single vehicle to the appropriate regulatory class, considering
320         'the vehicle's style, curb weight, shadow area, and domestic content, as well as
321         'the current boundary between cars and light trucks as major categories, and the
322         'scheme used to define regulatory [sub]classes of light trucks.
323         '
324         'If TestCurbWeight = NAN, then use Vehicles(iVeh).CurbWeight
325
326         ' Currently supported regulatory classes are:
327         '    1:  domestic passenger automobiles
328         '    2:  imported passenger automobiles
329         ' 3-10:  light trucks
330         '   11:  heavy light trucks regulated as a separate class
331         '
332         ' NOTE:  Unregulated vehicles are assigned regulatory class 0.
333         '
334         ' TODO:  add something to exclude heavy light trucks as appropriate
335         '
336         'Initialization
337         ' TODO:  resolve links to parameters for the CAFE program
338
339         If CAFE.Vehicles(iVeh).Model = "New Compact Pick-Up" Then
340             Debug.Write("")
341         End If
342         Dim Style As String = CAFE.Vehicles(iVeh).Style
343         Dim Weight As Double
344         If Double.IsNaN(TestCurbWeight) Then
345             Weight = CAFE.Vehicles(iVeh).CurbWeight
346         Else
347             Weight = TestCurbWeight
348         End If
349         Dim GVWR As Double = CAFE.Vehicles(iVeh).GVWR
350         Dim Area As Double = CAFE.Vehicles(iVeh).OverAllArea
351         Dim Origin As String = CAFE.Vehicles(iVeh).Origin
352         Dim DomesticContent As Double = CAFE.Vehicles(iVeh).US_Canadian_Content
353         Dim LightTruckDefn As Integer = CAFE.Scenarios.Scenarios(Scenario).LtDefinition(CAFE.GMYI(MY))
354         Dim HvyLtTruckIncl As Integer = CAFE.Scenarios.Scenarios(Scenario).HltInclusion(CAFE.GMYI(MY))
355         Dim CoveredHLT As Boolean = False
356         Dim HLT_FlatStd As Double = CAFE.Scenarios.Scenarios(Scenario).HLT_FlatSTD(CAFE.GMYI(MY)) ' flat standards for HLTs
357         Dim TruckClassType As String = CAFE.Scenarios.Scenarios(Scenario).LtRegClassAttribute
358         ' Dim nRegClass As Integer = CAFE.Parameters.nRegClass(CAFE.GMYI(MY)) ' total number reg. classes
359         Dim TruckClassMax() As Double = CAFE.Scenarios.Scenarios(Scenario).LtRegClassBoundaries(CAFE.GMYI(MY)) ' by model year
360         Dim LT_FunctionType As Integer = CAFE.Scenarios.Scenarios(Scenario).LtFunctionType(CAFE.GMYI(MY))
361         Dim RegClass As Integer = 0 ' regulatory class
362         Dim i As Integer = 0 ' counter variable
363         Dim AssignUsing As Double ' temporary variable for value of relevant attribute
364         Dim IsLightTruck As Boolean = True ' with LTDFN* as boolean, default to light truck
365         '
366         ' Processing
367
368         ' Decide whether vehicle is regulated with lighter vehicles.
369         Select Case HvyLtTruckIncl
370             Case 1
371                 CoveredHLT = CAFE.Vehicles(iVeh).HLTINCL1
372             Case 2
373                 CoveredHLT = CAFE.Vehicles(iVeh).HLTINCL2
374             Case 3
375                 CoveredHLT = CAFE.Vehicles(iVeh).HLTINCL3
376             Case 4
377                 CoveredHLT = CAFE.Vehicles(iVeh).HLTINCL4
378             Case 5
379                 CoveredHLT = CAFE.Vehicles(iVeh).HLTINCL5
380         End Select
```

```
381         If GVWR > 8500 AndAlso Not CoveredHLT Then
382             ' Vehicle is unregulated.
383             RegClass = 0
384         ElseIf GVWR > 8500 AndAlso HLT_FlatStd > 0 Then
385             ' Vehicle is a heavy light truck, and such vehicles are assigned to
386             ' a distinct regulatory class (11).
387             RegClass = 11
388         Else
389             ' Decide whether it's classified as a light truck (nonpass. auto.)
390             ' Decide whether it would be a light truck under default scheme.
391             ' In other words, default assumption is that new def'n is same as old def'n.
392             If Style = "Pickup" OrElse Style = "Utility" OrElse Style = "Van" OrElse Style
    = "Other" Then
393                 IsLightTruck = True
394             Else
395                 IsLightTruck = False
396             End If
397             ' Evaluate classification under new scheme (1-5 only; else leave at default).
398             ' TODO:  Consider making LTDFN* (and HvyLtTruckIncl) string variables so that
    user
399             ' can leave entry blank in order to indicate that default definition should be
    used.
400             Select Case LightTruckDefn
401                 Case 1
402                     IsLightTruck = CAFE.Vehicles(iVeh).LTDFN1
403                 Case 2
404                     IsLightTruck = CAFE.Vehicles(iVeh).LTDFN2
405                 Case 3
406                     IsLightTruck = CAFE.Vehicles(iVeh).LTDFN3
407                 Case 4
408                     IsLightTruck = CAFE.Vehicles(iVeh).LTDFN4
409                 Case 5
410                     IsLightTruck = CAFE.Vehicles(iVeh).LTDFN5
411             End Select
412             If Not IsLightTruck Then
413                 'If the vehicle is a car, assign it to either the domestic or import fleet.
414                 'Since we won't have domestic content for many vehicles, it might be easier
415                 'to go through the data and manually identify vehicles as belonging in
    either
416                 'the domestic or foreign fleet.  For MY02 vehicles already classified as
    cars,
417                 'this can come straight from the NHTSA CAFE database.  For vehicles that
    would
418                 'never be classified as cars, it can be ignored.  Light trucks that could
    be
419                 'classified will probably require close attention, though.
420                 '
421                 ' TODO:  Check inequality (is it > or >=?)
422                 '
423                 If Origin = "D" OrElse DomesticContent >= 0.75 Then
424                     RegClass = 1
425                 Else
426                     RegClass = 2
427                 End If
428             Else
429                 ' Begin with assumption that light trucks are a single regulatory class.
430                 RegClass = 3
431                 ' Assign vehicle to a light truck regulatory class.
432                 ' Use <= to test against upper boundaries.
433                 If TruckClassType = "None" OrElse Not (TruckClassMax(4) > TruckClassMax(3))
     OrElse LT_FunctionType = 9 Then
434                     ' Light trucks are covered as a single regulatory class.
435                     RegClass = 3
436                 Else
437                     If TruckClassType = "Area" Then AssignUsing = Area Else AssignUsing =
    Weight
438                     If AssignUsing <= TruckClassMax(3) Then
439                         RegClass = 3 '1st truck class
440                     ElseIf AssignUsing <= TruckClassMax(4) Then
441                         RegClass = 4 '2nd of multiple light truck classes
```

```
442                     ElseIf AssignUsing <= TruckClassMax(5) Then
443                         RegClass = 5 '3nd of multiple light truck classes
444                     ElseIf AssignUsing <= TruckClassMax(6) Then
445                         RegClass = 6 '4th of multiple light truck classes
446                     ElseIf AssignUsing <= TruckClassMax(7) Then
447                         RegClass = 7 '5th of multiple light truck classes
448                     ElseIf AssignUsing <= TruckClassMax(8) Then
449                         RegClass = 8 '6th of multiple light truck classes
450                     ElseIf AssignUsing <= TruckClassMax(9) Then
451                         RegClass = 9 '7th of multiple light truck classes
452                     Else
453                         RegClass = 10 '8th of 8 light truck classes
454                     End If ' End assignment to light truck subclass.
455                 End If ' End light truck classification.
456             End If ' End classification as car or light truck.
457         End If ' End classifiaction as unregulated or regulated.
458         '
459         'Termination
460         Return RegClass
461     End Function ' GetRegClass
462
463 End Class 'STD
464
```

```vb
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This class stores data that will be outputed in reports
'''
Public Class TechnologyStackClass
    Inherits CafeStackClass

    Public TechUsed() As Boolean

    Public Sub New()
        MyBase.New()
    End Sub

    Public Sub New(ByVal MY As Integer, ByVal veh As VehiclesClass)
        Me.New()
        CreateStack(MY, veh)
    End Sub

    Public Function CreateStack(ByVal MY As Integer, ByVal veh As VehiclesClass) As TechnologyStackClass
        Dim i As Integer

        Me.MY = MY
        If Not IsNothing(veh.TechUsed) Then
            ReDim Me.TechUsed(veh.TechUsed.Length - 1)
            For i = 0 To veh.TechUsed.Length - 1
                Me.TechUsed(i) = veh.TechUsed(i)
            Next i
        End If

        Return Me
    End Function

    Public Sub Copy(ByRef dst As TechnologyStackClass)
        Me.Copy(dst, Me)
    End Sub

    Public Shared Sub Copy(ByRef dst As TechnologyStackClass, ByRef src As TechnologyStackClass)
        Dim i As Integer

        dst = New TechnologyStackClass

        dst.MY = src.MY
        If Not IsNothing(src.TechUsed) Then
            ReDim dst.TechUsed(src.TechUsed.Length - 1)
            For i = 0 To src.TechUsed.Length - 1
                dst.TechUsed(i) = src.TechUsed(i)
            Next i
        End If
    End Sub

    Public Overloads Function Equals(ByRef tsc As TechnologyStackClass)
        Return Equals(Me, tsc)
    End Function

    Public Overloads Shared Function Equals(ByRef tsc1 As TechnologyStackClass, ByRef tsc2 As TechnologyStackClass) As Boolean
        Dim i As Integer

        If tsc1.TechUsed.Length <> tsc2.TechUsed.Length Then
            Return False
        End If

        For i = 0 To tsc1.TechUsed.Length - 1
            If tsc1.TechUsed(i) <> tsc2.TechUsed(i) Then
                Return False
            End If
        Next i

        ' all checks passed, return true
```

```
68          Return True
69      End Function
70
71      Public Function NotEquals(ByRef tsc As TechnologyStackClass)
72          Return Not Equals(Me, tsc)
73      End Function
74
75      Public Shared Function NotEquals(ByRef tsc1 As TechnologyStackClass, ByRef tsc2 As
        TechnologyStackClass) As Boolean
76          Return Not Equals(tsc1, tsc2)
77      End Function
78
79      Public Overrides Function ToArray() As Object()
80          Dim size As Integer = 1
81          Dim stack(size - 1) As Object
82
83          stack(0) = Me.TechUsed
84
85          Return stack
86      End Function
87
88  End Class
89
90  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
91  ' This class stores data per MFG loop
92  '''
93  Public Class TechnologiesStackClass
94      Inherits CafeStacksClass
95
96      Public Function ExpandTechUsed() As Boolean
97
98
99      End Function
100
101     '''
102     ' This function expands the TechUsed array...
103     '
104     Private Shared Function ExpandTechUsed(ByVal TechUsed() As Boolean,
105                                            ByVal TechGroup() As Integer) As Boolean()()
106         ''
107         'Dim i, j, k As Integer
108         'Dim ExpandedTechs()() As Boolean
109
110         'For i = 0 To TechGroup.Length - 1
111         '    If TechUsed(TechGroup(i)) Then
112         '        If IsNothing(ExpandedTechs) Then
113         '            k = 0
114         '        Else
115         '            k = ExpandedTechs.Length
116         '        End If
117
118         '        ReDim Preserve ExpandedTechs(k)
119         '        CopyArray(ExpandedTechs(k), TechUsed)
120
121         '        For j = 0 To TechGroup.Length - 1
122         '            ExpandedTechs(k)(TechGroup(j)) = False Or (j = i)
123         '        Next j
124         '    End If
125         'Next i
126
127         'If IsNothing(ExpandedTechs) Then
128         '    ReDim ExpandedTechs(0)
129         '    CopyArray(ExpandedTechs(0), TechUsed)
130         'End If
131
132         'Return ExpandedTechs
133     End Function
134
135
136
```

```
137
138     '''
139     ' Sets the TechUsed(TechIdx) element to TrueOrFalse starting at StartingStackIndex
140     '
141     Protected Sub SetTechUsed(ByVal StartingStackIndex As Integer, _
142                               ByVal TrueOrFalse As Boolean, _
143                               ByVal TechIdx As Integer)
144         '
145
146     End Sub
147
148
149
150
151
152     Public Overrides Function Push(ByVal tsc As CafeStackClass) As Boolean
153         ' compare the new element to the last element on the stack
154         If Me.MySize = 0 OrElse TechnologyStackClass.NotEquals(tsc, Me.MyStack(Me.MySize - ↙
        1)) Then
155             ' only add the new element if it is not the same as the last element on the ↙
        stack
156             '
157             ' create and copy the new VehicleStackClass
158             Dim entry As TechnologyStackClass
159             TechnologyStackClass.Copy(entry, tsc)
160             '
161             ' store it to the stack
162             Me.MyStack(Me.MySize) = entry
163             Me.MySize += 1
164
165             If Me.MySize >= Me.MyStack.Length Then
166                 ' array at max capacity; expand the array
167                 ReDim Preserve Me.MyStack(Me.MySize + (Me.CapacityIncrement - 1))
168             End If
169
170             Return True
171         Else
172             Return False
173         End If
174     End Function
175
176     Public Shadows Sub Copy(ByRef dst As TechnologiesStackClass)
177         Me.Copy(dst, Me)
178     End Sub
179
180     Public Shared Shadows Sub Copy(ByRef dst As TechnologiesStackClass, ByRef src As ↙
        TechnologiesStackClass)
181         Dim i As Integer
182
183         dst = New TechnologiesStackClass
184         CafeStacksClass.Copy(dst, src)
185
186         If Not IsNothing(src.MyStack) Then
187             ReDim dst.MyStack(src.MyStack.Length - 1)
188             For i = 0 To src.MyStack.Length - 1
189                 If Not IsNothing(src.MyStack(i)) Then
190                     TechnologyStackClass.Copy(dst.MyStack(i), src.MyStack(i))
191                 End If
192             Next i
193         End If
194     End Sub
195
196
197
198
199
200
201
202     '''
203     ' Returns the indexes of technologies within a same transmission superseding group.
```

```
204     '
205     Public Function GetTechTrnGroup() As Integer()
206         Dim tTrnGroup() As Integer = {8, 9, 10, 11, 12, 21}
207         Return tTrnGroup
208     End Function
209
210 End Class
211
212
```

```
Public Class TechnologyClass
    ' Single technology
    Public Name As String              ' Name of the technology
    Public Abbr As String              ' Abbreviation of the name of the technology
    Public FC_Low As Double            ' Fuel consumption improvements -- low end of range
    Public FC_High As Double           ' Fuel consumption improvements -- high end of range
    Public Cost_Low As Double          ' Cost estimate -- low end of range
    Public Cost_High As Double         ' Cost estimate -- high end of range
    Public YearAvailable As Integer    ' 4 digit year (e.g. 2008)
    Public Path1 As Boolean            ' Technology available in path 1
    Public Path2 As Boolean            ' Technology available in path 2
    Public Path3 As Boolean            ' technology available in path 3
    Public kWeight As Double           ' Weight coefficient
    Public PhaseIn As Double           ' Phase-in coefficient.

    ' Some technologies (specifically MSX) can be freely taken off a vehicle when added
    ' but only in the MY in which it was added
    Public Removable As Boolean        '
    Public _IsMaterialSubstitution As Boolean
    Public _IsDieselization As Boolean

    ''''' added for tech separation '''''
    Public TechType As String     ' the technology type

    Public Sub New()
        Removable = False
        _IsMaterialSubstitution = False
        _IsDieselization = False
    End Sub
    Public Sub New(ByRef SrcTech As TechnologyClass)
        Me.New()
        TechnologyClass.Copy(Me, SrcTech)
    End Sub

    Public Sub Copy(ByRef Dst As TechnologyClass)
        TechnologyClass.Copy(Dst, Me)
    End Sub

    Public Shared Sub Copy(ByRef Dst As TechnologyClass, ByRef Src As TechnologyClass)
        Dst.Name = Src.Name
        Dst.Abbr = Src.Abbr
        Dst.FC_Low = Src.FC_Low
        Dst.FC_High = Src.FC_High
        Dst.Cost_Low = Src.Cost_Low
        Dst.Cost_High = Src.Cost_High
        Dst.YearAvailable = Src.YearAvailable
        Dst.Path1 = Src.Path1
        Dst.Path2 = Src.Path2
        Dst.Path3 = Src.Path3
        Dst.kWeight = Src.kWeight
        Dst.PhaseIn = Src.PhaseIn
        Dst.TechType = Src.TechType
        Dst.Removable = Src.Removable
        Dst._IsMaterialSubstitution = Src._IsMaterialSubstitution
        Dst._IsDieselization = Src._IsDieselization
    End Sub
End Class  'Technology class

Public Class TechnologiesClass
    ' Combining all the technologies for one NAS class of vehicle
    Public Shared nTechVehicleClasses As Integer = 0    ' Number of classes of vehicles
    Public Shared nTranTechnologies As Integer = 0      ' Number of transmission
    modification techs
    Public Shared nEngTechnologies As Integer = 0       ' Number of engine modification
    techs
    Public Shared nAccLoadTechnologies As Integer = 0   ' Number of accessory load
    reduction techs
    Public Shared nMSMTechnologies As Integer = 0       ' Number of weight reduction techs
    Public Shared nMSPTechnologies As Integer = 0       ' Number of weight increase techs
    Public Shared nDLRTechnologies As Integer = 0       ' Number of other dynamic load
```

```
    reduction techs
68  Public Shared nAeroTechnologies As Integer = 0      ' Number of Aerodynamic load
    reduction techs
69  Public Shared nOtherTechnologies As Integer = 0     ' Number of other, undefined techs
70  Public Shared nTotalTechnologies As Integer = 0     ' Total number of technologies
71  Public Shared EngTechnologiesIdx(-1) As Integer     ' Which technologies are engine
    ones
72  Public Shared TranTechnologiesIdx(-1) As Integer    ' And Transmissions
73  Public Shared AccLoadTechnologiesIdx(-1) As Integer ' And accessory load
74  Public Shared MSMTechnologiesIdx(-1) As Integer     ' Indexes of material sub/weight
    reduce technologies
75  Public Shared MSPTechnologiesIdx(-1) As Integer     ' Indexes of weight increase
    technologies
76  Public Shared DLRTechnologiesIdx(-1) As Integer     ' Indexes of Dynamic load reduction
     technologies
77  Public Shared AeroTechnologiesIdx(-1) As Integer    ' Indexes of dynamic load reduction
     technologies
78  Public Shared OtherTechnologiesIdx(-1) As Integer   ' indexes of all other technologies
79  Public Shared TechIndexes As TechIndexesType        ' to allow lookup by tech type
80  Private Shared TechIndexesSet As Boolean = False
81
82  Public NRC_ClassName As String                      ' Small SUV, Minivan, etc.
83  Public nTechnologies As Integer                     ' Number of technologies
84  Public Technology() As TechnologyClass              ' The technologies
85
86  Public Sub New()
87      nTechnologies = 0
88      NRC_ClassName = String.Empty
89  End Sub
90  Public Sub New(ByVal VehClassName As String)
91      Me.New()
92      nTechVehicleClasses += 1
93      NRC_ClassName = VehClassName
94  End Sub
95
96  Public Sub Add(ByRef newTechnology As TechnologyClass)
97      'first copy raw data into 1 Technology array (we may want to remove that)
98      ReDim Preserve Technology(nTechnologies)
99      Technology(nTechnologies) = New TechnologyClass
100     newTechnology.Copy(Technology(nTechnologies))
101     ' If this is the first technology class being recorded,
102     ' then count the total number of technologies as well as the number within
103     ' the groups of technologies
104     ''''' added for tech separation ''''' <changed by jvs>
105     If TechnologiesClass.nTechVehicleClasses = 1 Then
106         Select Case newTechnology.TechType.ToUpper
107             Case "TRMOD"
108                 CountTechGroup(nTechnologies, nTranTechnologies, TranTechnologiesIdx)
109             Case "ENGMOD"
110                 CountTechGroup(nTechnologies, nEngTechnologies, EngTechnologiesIdx)
111             Case "ALR"
112                 CountTechGroup(nTechnologies, nAccLoadTechnologies,
    AccLoadTechnologiesIdx)
113             Case "MSM","WR"
114                 CountTechGroup(nTechnologies, nMSMTechnologies, MSMTechnologiesIdx)
115             Case "MSP"
116                 CountTechGroup(nTechnologies, nMSPTechnologies, MSPTechnologiesIdx)
117             Case "DLR"
118                 CountTechGroup(nTechnologies, nDLRTechnologies, DLRTechnologiesIdx)
119             Case "AERO"
120                 CountTechGroup(nTechnologies, nAeroTechnologies, AeroTechnologiesIdx)
121             Case Else
122                 ' catch-all case
123                 CountTechGroup(nTechnologies, nOtherTechnologies, OtherTechnologiesIdx)
124         End Select
125         If TechnologiesClass.TechIndexesSet = False Or TechnologiesClass.TechIndexesSet
     = True Then
126             TechnologiesClass.TechIndexesSet = True
127             Dim t As Integer = nTechnologies
128             Select Case newTechnology.Abbr
```

```
129                     Case "EFR"
130                         TechIndexes.tEngEFR = t
131                     Case "LUB"
132                         TechIndexes.tEngLUB = t
133                     Case "OHC"
134                         TechIndexes.tEngOHC = t
135                     Case "VVT"
136                         TechIndexes.tEngVVT = t
137                     Case "VVLT"
138                         TechIndexes.tEngVVLT = t
139                     Case "DISP"
140                         TechIndexes.tEngDSP = t
141                     Case "EAI"
142                         TechIndexes.tEngEAI = t
143                     Case "SIDI"
144                         TechIndexes.tEngSIDI = t
145                     Case "SUP"
146                         TechIndexes.tEngSUP = t
147                     Case "5SP"
148                         TechIndexes.tTrn5SP = t
149                     Case "CVT"
150                         TechIndexes.tTrnCVT = t
151                     Case "ASL"
152                         TechIndexes.tTrnASL = t
153                     Case "6SP"
154                         TechIndexes.tTrn6SP = t
155                     Case "AER"
156                         TechIndexes.tVehAERO = t
157                     Case "ROLL"
158                         TechIndexes.tVehROLL = t
159                     Case "LDB"
160                         TechIndexes.tVehLDB = t
161                     Case "FAD"
162                         TechIndexes.tVehFAD = t
163                     Case "IVT"
164                         TechIndexes.tEngIVT = t
165                     Case "CVA"
166                         TechIndexes.tEngCVA = t
167                     Case "VCR"
168                         TechIndexes.tEngVCR = t
169                     Case "AST"
170                         TechIndexes.tTrnAST = t
171                     Case "ACVT"
172                         TechIndexes.tTrnACV = t
173                     Case "42V"
174                         TechIndexes.tVeh42V = t
175                     Case "ISG"
176                         TechIndexes.tVehISG = t
177                     Case "EPS"
178                         TechIndexes.tVehEPS = t
179                     Case "DSL"
180                         TechIndexes.tEngDSL = t
181                     Case "MS1", "WGT"
182                         TechIndexes.tMS_MS1 = t
183                     Case "MS2"
184                         TechIndexes.tMS_MS2 = t
185                     Case "MS3"
186                         TechIndexes.tMS_MS3 = t
187                     Case "MSX"
188                         TechIndexes.tMS_MSX = t
189                     Case "HEV"
190                         TechIndexes.tTrnHEV = t
191                     Case Else
192                         MsgBox("Unknown Abbrev: " & newTechnology.Abbr)
193                 End Select
194             End If
195
196             nTotalTechnologies += 1
197         End If
198         nTechnologies += 1
```

```
199     End Sub
200
201     ''''' added for tech separation
202     Private Sub CountTechGroup(ByVal newTechIdx As Integer, _
203                                ByRef TechGroupCount As Integer, _
204                                ByRef TechGroupIdxList() As Integer)
205         ' Allocte space on the index list for this technology group
206         ' add the new tech to the group and bump the count for that group
207         Dim len As Integer
208         If IsNothing(TechGroupIdxList) Then
209             len = 0
210         Else
211             len = TechGroupIdxList.Length
212         End If
213         ReDim Preserve TechGroupIdxList(len)
214         TechGroupIdxList(len) = newTechIdx
215         Array.Sort(TechGroupIdxList)            ' Keep sorted to allow quick binarysearch ↙
    (jvs 16-aug-04)
216         TechGroupCount += 1
217     End Sub
218
219     Public Sub Copy(ByRef Dst As TechnologiesClass)
220         TechnologiesClass.Copy(Dst, Me)
221     End Sub
222     Public Shared Sub Copy(ByRef Dst As TechnologiesClass, ByRef Src As TechnologiesClass)
223         Dim i As Integer
224         Dst.NRC_ClassName = Src.NRC_ClassName
225         Dst.nTechnologies = Src.nTechnologies
226         ReDim Dst.Technology(Src.nTechnologies - 1)
227         For i = 0 To Src.nTechnologies - 1
228             Dst.Technology(i) = New TechnologyClass
229             Src.Technology(i).Copy(Dst.Technology(i))
230         Next
231     End Sub
232
233     Public Function GetNumTechnologies() As Integer
234         Return nTechnologies
235     End Function
236
237     Public Function GetCost(ByVal TechIdx As Integer) As Double
238         Dim cost As Double = Double.NaN
239         Select Case CAFE.TechnologyCostEstimates
240             Case CAFE.LowAverageHigh.Low
241                 cost = Technology(TechIdx).Cost_Low
242             Case CAFE.LowAverageHigh.High
243                 cost = Technology(TechIdx).Cost_High
244             Case CAFE.LowAverageHigh.Average
245                 cost = (Technology(TechIdx).Cost_High + Technology(TechIdx).Cost_Low) / 2
246         End Select
247         Return cost
248     End Function
249
250     Public Function GetImprv(ByVal TechIdx As Integer) As Double
251         Dim imprv As Double = Double.NaN
252         Select Case CAFE.FuelConsumptionSavingsEstimates
253             Case CAFE.LowAverageHigh.Low
254                 imprv = Technology(TechIdx).FC_Low
255             Case CAFE.LowAverageHigh.High
256                 imprv = Technology(TechIdx).FC_High
257             Case CAFE.LowAverageHigh.Average
258                 imprv = (Technology(TechIdx).FC_High + Technology(TechIdx).FC_Low) / 2
259         End Select
260         Return imprv
261     End Function
262
263     ''''' added for tech separation '''''
264 #Region "''''' added for tech separation '''''"
265
266     Public Function GetNumTranTechnologies() As Integer
267         Return nTranTechnologies
```

```
268     End Function
269
270     Public Function GetNumEngTechnologies() As Integer
271         Return nEngTechnologies
272     End Function
273
274     Public Function GetNumAccLoadTechnologies() As Integer
275         Return nAccLoadTechnologies
276     End Function
277
278     Public Function GetNumWgtTechnologies() As Integer
279         Return nMSMTechnologies
280     End Function
281
282     Public Function GetNumDLRTechnologies() As Integer
283         Return nDLRTechnologies
284     End Function
285
286     Public Function GetNumOtherTechnologies() As Integer
287         Return nOtherTechnologies
288     End Function
289
290 #End Region
291
292 End Class   ' Technologies class
293 Public Structure TechIndexesType
294     Public tEngEFR As Integer           ' Engine Friction Reduction
295     Public tEngLUB As Integer           ' Low Friction Lubricants
296     Public tEngOHC As Integer           ' Multi-valve, overhead camshaft
297     Public tEngVVT As Integer           ' Variable Value Timing
298     Public tEngVVLT As Integer          ' Variable Valve Lift & Timing
299     Public tEngDSP As Integer           ' Cylinder deactivation (displacement)
300     Public tEngEAI As Integer           ' Engine accessory improvement
301     Public tEngSIDI As Integer          ' Spark-ignited direct injection
302     Public tEngSUP As Integer           ' Engine supercharging and downsizing
303     Public tEngDSL As Integer           ' Engine dieselization
304
305     ' Production-intent Transmission Technologies
306     Public tTrn5SP As Integer           ' 5 speed automatic transmission
307     Public tTrnCVT As Integer           ' Continuously variable transmission
308     Public tTrnASL As Integer           ' Automatic w/aggressive shift logic
309     Public tTrn6SP As Integer           ' 6 speed automatic transmission
310
311     ' Production-intent Vehicle Technologies
312     Public tVehAERO As Integer          ' Aero drag reduction
313     Public tVehROLL As Integer          ' Improved rolling resistance
314     Public tVehLDB As Integer           ' Low-drag brakes
315     Public tVehFAD As Integer           ' Front axle disconnect (for 4WD)
316
317     ' Material Substitution (to change vehicle weight)
318     Public tMS_MS1 As Integer           ' First level of weight reduction
319     Public tMS_MS2 As Integer           ' Second level of weight reduction
320     Public tMS_MS3 As Integer           ' Third level of weight reduction
321     Public tMS_MSX As Integer           ' Adding weight
322
323     ' Emerging Engine Technologies
324     Public tEngIVT As Integer           ' Intake valve throttling
325     Public tEngCVA As Integer           ' Camless value actuation
326     Public tEngVCR As Integer           ' Variable Compression Ratio
327
328     ' Emerging Transmission Technologies
329     Public tTrnAST As Integer           ' Automatic shift manual transmission (AST/AMT)
330     Public tTrnACV As Integer           ' Advanced Continuously variable transmission
331     Public tTrnHEV As Integer           ' Midrange High Efficiency Vehicle (Hybrid)
332
333     ' Emerging Vehicle Technologies
334     Public tVeh42V As Integer           ' 42 Volt electrical systems
335     Public tVehISG As Integer           ' Integrated Starter/Generator
336     Public tVehEPS As Integer           ' Electrical power steering
337 End Structure
```

```
1  Public Class TransmissionsClass
2
3  #Region "TransmissionsData"
4      Public Valid As Boolean  ' Run-time generated
5      Public ReasonForBeingInvalid As String
6      Public MyIndexNumber As Integer     ' Run-time generated
7
8      Public Shared nMostModels As Integer = 0     ' Stores the number of models for the
9      '                                            ' transmission with the most models
10
11     Public Transmission_Code As Integer
12     Public Manufacturer As String
13     Public MfgIdx As Integer
14     Public Name As String
15     Public Origin As String
16     Public Type As String
17     Public NumberForwardGears As Integer
18     Public Control As String
19     Public Logic As String
20     Public GearRatioFirstGear As Double      '''''''''''''''
21     Public GearRatioSecondGear As Double     '''''''''''''''
22     Public GearRatioThirdGear As Double      '''''''''''''''
23     Public GearRatioFourthGear As Double     '''''''''''''''
24     Public GearRatioFifthGear As Double      '''''''''''''''
25     Public GearRatioSixthGear As Double      '''''''''''''''
26     Public ReverseGearRatio As Double        '''''''''''''''
27     Public MaxGearRatio As Double
28     Public MinGearRatio As Double
29     Public TCRatio As Double
30     Public AxleRatio As Double       '''''''''''''''
31     Public TCLockupBypass As Boolean
32     Public TransmissionFluidSpecification As String
33     Public TransmissionFluidViscosity As String
34     Public TransmissionNotes As String
35
36     Public TechOver5SP As CAFE.TechOverRide      ' field for the 5SP col
37     Public TechOver6SP As CAFE.TechOverRide      ' field for the 6SP col
38     Public TechOverASL As CAFE.TechOverRide
39     Public TechOverCVT As CAFE.TechOverRide
40     Public TechOverAST As CAFE.TechOverRide
41     Public TechOverACVT As CAFE.TechOverRide
42     Public TechOverHEV As CAFE.TechOverRide
43
44     Public ParentIndex As Integer   ' Index to the parent transmission, if there is one
45
46     Public nCompleteModels As Integer        ' Number of models in complete list
47     Public CompleteModelsList() As Integer   ' Complete list of models which use this engine
48     Public nModels As Integer                ' Number of valid models that use this
       transmission
49     Public ModelsList() As Integer           ' List of valid models that use this
       transmission
50     Public TechCost As Double                ' Technology cost applied to this transmission.
51     Public TechImprv As Double               ' Technology improvements applied to this
       transmission.
52     Public TechUsed() As Boolean             ' Transmission Techs used by this transmission
53 #End Region
54
55     Public Sub New()
56         ParentIndex = -1     ' set parent index to -1 to indicate that this
57         '                    ' transmission does not have a parent transmission
58         Valid = False
59         nModels = 0
60         TechCost = 0
61         TechImprv = 1
62         ReDim TechUsed(-1)
63     End Sub
64     Public Shared Function Clone(ByVal srcIdx As Integer, ByVal VehList() As Integer) As
       Integer
65         Dim tIdx As Integer = CAFE.Transmissions.Length
66         Debug.WriteLine("Adding new transmission (# " + tIdx.ToString + ")")
```

```
67
68          ' Check if the number of vehicles for this trans is the same as the 'most'
69          ' if so, we'll want to reset 'most' after our cloning.
70          Dim RecalcMost As Boolean = (CAFE.Transmissions(srcIdx).nCompleteModels =
        nMostModels)
71
72          ReDim Preserve CAFE.Transmissions(tIdx)
73          TransmissionsClass.Copy(CAFE.Transmissions(tIdx), CAFE.Transmissions(srcIdx))
74          ' Now remove vehicles being split off from source engine
75          ' Do CompleteModelsList
76          For v As Integer = 0 To VehList.Length - 1
77              Dim vv As Integer = 0
78              Do
79                  If VehList(v) = CAFE.Transmissions(srcIdx).CompleteModelsList(vv) Then
80                      ' Remove this vehicle
81                      Dim r As Integer
82                      For r = vv + 1 To CAFE.Transmissions(srcIdx).CompleteModelsList.Length
        - 1
83                          CAFE.Transmissions(srcIdx).CompleteModelsList(r - 1) = CAFE.
        Transmissions(srcIdx).CompleteModelsList(r)
84                      Next
85                      r = CAFE.Transmissions(srcIdx).CompleteModelsList.Length - 2
86                      ReDim Preserve CAFE.Transmissions(srcIdx).CompleteModelsList(r)
87                      CAFE.Transmissions(srcIdx).nCompleteModels = r + 1
88                  Else
89                      vv = vv + 1
90                  End If
91              Loop While vv < CAFE.Transmissions(srcIdx).CompleteModelsList.Length
92              ' Add the vehicle to the new transmission
93              CAFE.Transmissions(tIdx).AddUser(VehList(v))
94          Next v
95          ' Now populate ModelsList for both
96          CAFE.Transmissions(srcIdx).CreateModelsList()
97          CAFE.Transmissions(tIdx).CreateModelsList()
98
99          If RecalcMost Then
100             nMostModels = 0
101             For t As Integer = 0 To CAFE.Transmissions.Length - 1
102                 nMostModels = Math.Max(nMostModels, CAFE.Transmissions(t).nCompleteModels)
103             Next
104         End If
105         Return tIdx
106     End Function
107
108     Public Sub Merge(ByVal FromTransmissionIdx As Integer)
109         ' Merge an engine... assumes that the caller knows that the two engines are
110         ' indentical or not.
111         For Each vidx As Integer In CAFE.Transmissions(FromTransmissionIdx).ModelsList
112             Me.AddUser(vidx)
113         Next
114     End Sub
115     Public Sub Copy(ByRef Dst As TransmissionsClass)
116         TransmissionsClass.Copy(Dst, Me)
117     End Sub
118     Public Shared Sub Copy(ByRef Dst As TransmissionsClass, ByRef Src As
        TransmissionsClass)
119         Dim i As Integer
120
121         Dst = New TransmissionsClass
122
123         Dst.Valid = Src.Valid
124         Dst.MyIndexNumber = Src.MyIndexNumber
125         Dst.ParentIndex = Src.ParentIndex
126
127         Dst.Transmission_Code = Src.Transmission_Code
128         Dst.Manufacturer = Src.Manufacturer
129         Dst.Name = Src.Name
130         Dst.Type = Src.Type
131         Dst.Origin = Src.Origin
132         Dst.NumberForwardGears = Src.NumberForwardGears
```

```
133         Dst.Control = Src.Control
134         Dst.Logic = Src.Logic
135         Dst.GearRatioFirstGear = Src.GearRatioFirstGear
136         Dst.GearRatioSecondGear = Src.GearRatioSecondGear
137         Dst.GearRatioThirdGear = Src.GearRatioThirdGear
138         Dst.GearRatioFourthGear = Src.GearRatioFourthGear
139         Dst.GearRatioFifthGear = Src.GearRatioFifthGear
140         Dst.GearRatioSixthGear = Src.GearRatioSixthGear
141         Dst.ReverseGearRatio = Src.ReverseGearRatio
142         Dst.MaxGearRatio = Src.MaxGearRatio
143         Dst.MinGearRatio = Src.MinGearRatio
144         Dst.TCRatio = Src.TCRatio
145         Dst.AxleRatio = Src.AxleRatio
146         Dst.TCLockupBypass = Src.TCLockupBypass
147         Dst.TransmissionFluidSpecification = Src.TransmissionFluidSpecification
148         Dst.TransmissionFluidViscosity = Src.TransmissionFluidViscosity
149         Dst.TransmissionNotes = Src.TransmissionNotes
150         Dst.TechOver5SP = Src.TechOver5SP
151         Dst.TechOver6SP = Src.TechOver6SP
152         Dst.TechOverASL = Src.TechOverASL
153         Dst.TechOverCVT = Src.TechOverCVT
154         Dst.TechOverAST = Src.TechOverAST
155         Dst.TechOverACVT = Src.TechOverACVT
156         Dst.TechOverHEV = Src.TechOverHEV
157
158         Dst.nCompleteModels = Src.nCompleteModels
159         ReDim Dst.CompleteModelsList(Dst.nCompleteModels - 1)
160         For i = 0 To Dst.nCompleteModels - 1
161             Dst.CompleteModelsList(i) = Src.CompleteModelsList(i)
162         Next
163
164         Dst.nModels = Src.nModels
165         ReDim Dst.ModelsList(Src.nModels - 1)
166         For i = 0 To Src.nModels - 1
167             Dst.ModelsList(i) = Src.ModelsList(i)
168         Next
169         Dst.TechCost = Src.TechCost
170         Dst.TechImprv = Src.TechImprv
171         Dst.MfgIdx = Src.MfgIdx
172         If IsNothing(Src.TechUsed) = False Then
173             ReDim Dst.TechUsed(Src.TechUsed.Length - 1)
174             For i = 0 To Src.TechUsed.Length - 1
175                 Dst.TechUsed(i) = Src.TechUsed(i)
176             Next
177         End If
178     End Sub
179     Public Sub AddUser(ByVal VehIndex As Integer)
180         ' Add a vehicle (index to vehicles) as a vehicle which uses this transmission
181         ReDim Preserve CompleteModelsList(nCompleteModels)      ' need to preserve the
    array each time  <ms>
182         CompleteModelsList(nCompleteModels) = VehIndex
183         nCompleteModels = nCompleteModels + 1
184
185         If nCompleteModels > nMostModels Then
186             nMostModels = nCompleteModels
187         End If
188     End Sub
189
190     '''
191     ' CreateModelsList
192     ' This sub populates the ModelsList from valid vehicles in the CompleteModelsList
193     '
194     Public Sub CreateModelsList()
195         nModels = 0
196         ReDim ModelsList(-1)
197         If Me.Valid Then
198             For v As Integer = 0 To CompleteModelsList.Length - 1
199                 If CAFE.Vehicles(CompleteModelsList(v)).IsValid Then
200                     ' got a valid one
201                     ReDim Preserve ModelsList(ModelsList.Length)
```

```
202                     ModelsList(ModelsList.Length - 1) = CompleteModelsList(v)
203                 End If
204             Next
205             nModels = ModelsList.Length
206             Dim NumTechnologies = CAFE.Technologies.GetNumTechnologies
207         End If
208     End Sub
209
210     '''
211     ' Clears the CompleteModelsList() and ModelsList() arrays for the new run.
212     '
213     Public Sub ClearModelLists()
214         ReDim Me.CompleteModelsList(-1)
215         Me.nCompleteModels = 0
216         ReDim Me.ModelsList(-1)
217         Me.nModels = 0
218     End Sub
219
220 #Region "TechOverRide Support Routines"
221     Public Function IsTechOverRideForce(ByVal TechIdx As Integer) As Boolean
222         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Force
223     End Function
224     Public Function IsTechOverRideForce(ByVal TechAbbreviation As String) As Boolean
225         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Force
226     End Function
227     Public Function IsTechOverRideSkip(ByVal TechIdx As Integer) As Boolean
228         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Skip
229     End Function
230     Public Function IsTechOverRideSkip(ByVal TechAbbreviation As String) As Boolean
231         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Skip
232     End Function
233     Public Function IsTechOverRideUsed(ByVal TechIdx As Integer) As Boolean
234         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Used
235     End Function
236     Public Function IsTechOverRideUsed(ByVal TechAbbreviation As String) As Boolean
237         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Used
238     End Function
239     Public Function IsTechOverRideNone(ByVal TechIdx As Integer) As Boolean
240         Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.None
241     End Function
242     Public Function IsTechOverRideNone(ByVal TechAbbreviation As String) As Boolean
243         Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.None
244     End Function
245     '''
246     ' Get the CAFE.TechOverRIde setting.
247     ' This is rather brute-force with a complete lack of elegance,
248     ' maybe we'll clean this up someday -- but don't count on it
249     ' jvs 15 oct 04
250     Public Function GetTechOverRide(ByVal TechIdx As Integer) As CAFE.TechOverRide
251         Dim Abbr As String
252         ' Get the abbreviation using NRC Vehicle Class 0 (the abbrv are the same for all)
253         Abbr = CAFE.Technologies.VehicleClasses(0).Technology(TechIdx).Abbr
254         Return GetTechOverRide(Abbr)
255     End Function
256     Public Function GetTechOverRide(ByVal TechAbbreviation As String) As CAFE.TechOverRide
257         Dim RtnCode As CAFE.TechOverRide = CAFE.TechOverRide.None
258         Select Case TechAbbreviation
259             Case "5SP"
260                 RtnCode = Me.TechOver5SP
261             Case "6SP"
262                 RtnCode = Me.TechOver6SP
263             Case "ASL"
264                 RtnCode = Me.TechOverASL
265             Case "CVT"
266                 RtnCode = Me.TechOverCVT
267             Case "AST"
268                 RtnCode = Me.TechOverAST
269             Case "ACVT"
270                 RtnCode = Me.TechOverACVT
271             Case "HEV"
```

```
272                     RtnCode = Me.TechOverHEV
273             End Select
274             Return RtnCode
275         End Function
276 #End Region
277 End Class
278
```

```
Public Class VehiclesClass

#Region " Model Data "
    Public Structure VehiclePredInfo
        Public Populated As Boolean
        Public Type As String           ' "P" for predecessor, "L" as legacy
        Public PredIdx As Integer
        Public TechUsed() As Boolean
        Public FE_Before As Double
        Public FE_After As Double
        Public TechCost As Double
        Public TechImprv As Double
        Public OrigCW As Double
        Public NewCW As Double
        Public MY As Integer            ' model year we're doing this...
    End Structure

    Public MissingData As Boolean       ' determines whether this vehicle has incomplete or missing data
    Public Valid As Boolean
    Public ValidOverAll As Boolean      ' Valid over-all
    Public ValidForScenario As Boolean  ' Valid for the scenario
    Public ValidForMY As Boolean        ' Valid for the Model Year
    Public ReasonForBeingInvalid As String

    Public MyIndexNumber As Integer     ' run-time value
    Public MinYear As Integer           ' first year with a fuel economy value

    Public Number As Integer
    Public Manufacturer As String
    Public MfgIdx As Integer            ' so we can find our MFG in CAFE
    Public Model As String
    Public NamePlate As String      '''''''''''''''
    Public NamePlateIdx As Integer      ' in CAFE.Mfgs(n).NamePlates
    Private _FuelEconomy As Double
    Public InitialFuelEconomy As Double
    Private _ActualFE_FFVs As Double  '''''''''''''''
    Public EngineCode As Integer    ' Index to engine type
    Public TrueEngineCode As Integer    ' Adjusted to the realities of life
    Public TransmissionCode As Integer
    Public TrueTransmissionCode As Integer
    Public Origin As String
    Public US_Canadian_Content As Double
    'Public FinalAssemblyCity As String
    'Public FinalAssemblyState As String
    'Public FinalAssembly As String
    'Public GeneralNotes As String
    Public PredecessorFromFile As Integer
    Public Predecessor As Integer               ' adjusted to 'true' value
    Public PredecessorUsed As Boolean           ' initially false. Once the pred info used, set to true.
    Public PredInfo As VehiclePredInfo          ' Predecessor Info
    Public LastFreshening As Integer
    Public NextFreshening As Integer
    Public LastRedesign As Integer
    Public NextRedesign As Integer
    Public EmploymentHoursPerVehicle As Double  '''''''''''''''
    'Public PlanningNotes As String
    Public nSalesYears As Integer
    Public Sales() As Double
    Public InitialSales() As Double
    'Public SalesNotes As String
    Public nMSRPYears As Integer
    Public MSRP() As Double
    Public InitialMSRP() As Double
    'Public MSRPNotes As String
    Public nPriceYears As Integer
    Public Price() As Double
    Public InitialPrice() As Double
    'Public PriceNotes As String
```

```
69      Public HybridType As String
70      Public VoltageOrPressure As Double
71      Public EnergyStorageCapacity As Double
72      Public BatteryType As String
73      Public EnergyTransfer As String
74      Public BrakingEnergyRecovery As Double
75      Public ShareMaximumPower As Double
76      'Public HybridizationNotes As String
77      Public Style As String
78      Public VehClass As String
79      Public VehStructure As String
80      Public Drive As String
81      Public FinalDriveRatio As Double
82      Public EngineSpeedOverVehSpeed As Double
83      'Public FrontAxleViscosity As String
84      'Public RearAxleViscosity As String
85      'Public OverAllLength As Double '[KG:  changed 01/24/05 for footprint-based system]
86      'Public OverAllWidth As Double '[KG:  changed 01/24/05 for footprint-based system]
87      Public Wheelbase As Double '[KG:  changed 01/24/05 for footprint-based system]
88      Public TrackWidthFront As Double '[KG:  changed 01/24/05 for footprint-based system]
89      Public TrackWidthRear As Double '[KG:  changed 01/24/05 for footprint-based system]
90      Public OverAllArea As Double
91      Public OverAllHeight As Double
92      'Public GroundClearance As Double
93      'Public FrontAxleClearance As Double
94      'Public RearAxleClearance As Double
95      'Public AngleOfApproach As Double
96      'Public BreakoverAngle As Double
97      'Public AngleOfDeparture As Double
98      'Public CenterOfGravityHeight As Double
99      Public CurbWeight As Double
100     Public InitialCurbWeight As Double
101     Public TestWeight As Double
102     Public PAUSetting As Double
103     Public GVWR As Double
104     Public StandardTowingCapacity As Double
105     Public MaxTowingCapacity As Double
106     Public Payload As Double
107     Public MinSeating As Integer
108     Public MaxSeating As Integer
109     'Public FrontRowSeating As Integer
110     'Public CargoVolBehindFirstRow As Double
111     'Public SecondRowSeating As Integer
112     'Public SecondRowFlatCapability As String
113     'Public CargoVolBehindSecondRow As Double
114     'Public ThirdRowSeating As Integer
115     'Public ThirdRowFlatCapability As String
116     'Public CargoVolBehindThirdRow As Double
117     'Public EnclosedVolume As Double
118     'Public StandardPassengerVolume As Double
119     'Public MaxPassengerVolume As Double
120     'Public CargoVolumeIndex As Double
121     'Public StandardEnclosedCargoVolume As Double    '''''''''''''''
122     'Public MaxEnclosedCargoVolume As Double         '''''''''''''''
123     'Public OpenBoxAreaLength As Double
124     'Public MinOpenBoxAreaWidth As Double
125     'Public MaxOpenBoxAreaWidth As Double
126     Public OpenBoxArea As Double
127     Public OpenBoxHeight As Double
128     Public FuelCapacity As Double
129     'Public TireRollingResistance As Double
130     'Public FrontalArea As Double
131     'Public RecommendedTirePressure As Double    '''''''''''''''
132     'Public TireStaticLoadedRadius As Double     '''''''''''''''
133     'Public TireTreadWidth As Double             '''''''''''''''
134     'Public TireDeflection As Double             '''''''''''''''
135     'Public AerodynamicDragCoefficient As Double
136     'Public VehicleNotes As String
137     'Public EPA_Class As String
138     'Public EPA_Certification As Integer
```

```
139     'Public LEV_Class As String
140     'Public RegulatoryNotes As String
141     Public StageAnalysisFE() As Double
142     Public StageAnalysisCW() As Double
143     Public StageAnalysisSales() As Double
144
145     Public LTDFN1 As Boolean
146     Public LTDFN2 As Boolean
147     Public LTDFN3 As Boolean
148     Public LTDFN4 As Boolean
149     Public LTDFN5 As Boolean
150     Public HLTINCL1 As Boolean
151     Public HLTINCL2 As Boolean
152     Public HLTINCL3 As Boolean
153     Public HLTINCL4 As Boolean
154     Public HLTINCL5 As Boolean
155     Public MOBILE6Class As String       ' Even though there is a column for this variable in the
156     '                                   ' input file, this variable is determined during run-time
157     Public SafetyClass As String
158     Public Percent2DR As Double
159     Public MarketSegment_VOLPE As Integer
160     Public MarketSegment_AutoNews As Integer
161     'Public MarketNest As Integer       ' replaced by market segment
162     Public TechOverROLL As CAFE.TechOverRide
163     Public TechOverLDB As CAFE.TechOverRide
164     Public TechOverFAD As CAFE.TechOverRide
165     Public TechOverEPS As CAFE.TechOverRide
166     Public TechOverEAI As CAFE.TechOverRide
167     Public TechOverAER As CAFE.TechOverRide
168     Public TechOver42V As CAFE.TechOverRide
169     Public TechOverISG As CAFE.TechOverRide
170     Public TechOverMS1 As CAFE.TechOverRide
171     Public TechOverMS2 As CAFE.TechOverRide
172     Public TechOverMS3 As CAFE.TechOverRide
173     Public TechOverMSX As CAFE.TechOverRide
174
175     Public NRC_Class As String                  ' the NRC technology class for this vehicle
176     Public NRC_ClassIdx As Integer              ' the NRC Technology class as an index
177
178     Public RegClass As Integer                  ' vehicle's current regulatory class
179     Public RegCost As Double                    ' regulatory costs currently assigned to vehicle by AllocateCosts
180     'Public TechCost As Double                   ' technology costs currently incurred by vehicle
181     Public TechCostByMY() As Double             ' technology costs each MY
182     Public TechCostVehTechnologies As Double    ' Tech costs for vehicle (not engine or transmission) technologies.
183     Public TechImprvVehTechnologies As Double   ' Tech imprv for vehicle (ditto)
184     Public ModeledBaselineSales() As Double     ' baseline sales (by MY) forecast using market share model
185     Public Utility As Double                    ' current utility of vehicle as calculated by market share model
186     Public Elasticity(,) As Double              ' NOTE:  may be able to omit this if carried in Paremeters
187
188     Public SalesOverFE As Double                ' For CAFE Calc. Calculated at the start of each MY processing
189     Public TechClass As Integer                 ' Technology class for this vehicle
190     Public TechEnabled() As Boolean             ' if true, then the technology is available (reinitialized for each MY)
191     Public TechApplies() As Boolean             ' if true, then the technology is available (initialed once when model starts)
192     Public TechUsed() As Boolean                ' set true if the technology has been used on this vehicle
193     Public TechReserved() As Boolean            ' stores any tech that the model has seen and estimated, and is likely to apply
194     Public TechAppliedByModel() As Boolean      ' set true if the model applied the
```

```
    tech.
195     Public TechUsedByMY()() As Boolean              ' same as tech used, except it is by ↙
    MY
196     Public TechIgnored() As Boolean                 ' short-term (once per MY) means of  ↙
    turning off a technology
197     Public TechMS2Enablable As Boolean              ' Whether to enable MS2 when MS1 is  ↙
    installed
198     Public TechMS3Enablable As Boolean              ' Whether to enable MS3 when MS2 is  ↙
    installed
199     Public TechUsedStack As TechnologiesStackClass  ' stores all technologies            ↙
    incrementally applied to this veh for a given MY
200     Public ScaleA5 As Boolean                       ' Whether or not to scale improvements ↙
     in going from M4 to A5
201     Public ScaleA6 As Boolean                       ' Whether or not to scale improvements ↙
     in going from M5 to A6
202     Public ScaleCVT As Boolean                      ' Whether or not to scale improvements ↙
     in going from M5 or M6 to CVT
203     Public Fleet() As Double                        ' stores the fleet for this vehicle by ↙
     CY (this value is updated with each MY)
204     Public VMT() As Double                          ' stores the VMT for this vehicle by ↙
    CY (this value is updated with each MY)
205     Public CPM() As Double                          ' stores the CPM for this vehicle by ↙
    CY (this value is updated with each MY)
206     Public Fatals() As Double                       ' stores the fatalaties for this     ↙
    vehicle by CY (this value is updated with each MY)
207     Public Gallons() As Double                      ' stores the fuel gallons of this    ↙
    vehicle
208     Public fuelTax() As Double                      ' stores the fuel tax revenues of this ↙
     vehicle
209     Public discFuelTax() As Double                  ' stores the discounted fuel tax     ↙
    revenues of this vehicle
210     Public preTaxFuel() As Double                   ' stores the pre-tax fuel expenditures ↙
     of this vehicle
211     Public discPreTaxFuel() As Double               ' stores the discounted pre-tax fuel ↙
    expenditures of this vehicle
212     Public driveSurplus() As Double
213     Public initialDriveSurplus() As Double
214     Public discDriveSurplus() As Double
215     Public refuelSurplus() As Double
216     Public initialRefuelSurplus() As Double
217     Public discRefuelSurplus() As Double
218
219     Public Shared xRunSafetyCalcs As Boolean        ' specifies whether to run safety    ↙
    calcs and generate its output
220
221     '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
222     ' Properties to hide Fuel Economy.
223     ' If _ActualFE_FFVs is set, then we'll use IT as the fuel economy for all model       ↙
    calculations,
224     ' otherwise, we'll use _FuelEconomy.
225     Public Property FuelEconomy() As Double
226         Get
227             If Me._ActualFE_FFVs > 0 Then
228                 Return Me._ActualFE_FFVs
229             Else
230                 Return Me._FuelEconomy
231             End If
232         End Get
233         Set(ByVal Value As Double)
234             If Me._ActualFE_FFVs > 0 Then
235                 Me._ActualFE_FFVs = Value
236             Else
237                 Me._FuelEconomy = Value
238             End If
239         End Set
240     End Property
241     '
242     ' TechCost property to amalgamate the various techcosts (Veh, Eng, Tran & Nameplate)
243     Public ReadOnly Property TechCost() As Double
244         ' Alternate means: jvs 09-mar-05 -- use tech used for the veh and individual tech ↙
```

```
    cost from technologies
        Get
            Dim tc As Double = 0
            Dim CWCoeff As Double
            Dim DeltaCW As Double
            Dim OldCW As Double = Me.InitialCurbWeight
            For t As Integer = 0 To TechnologiesClass.nTotalTechnologies - 1
                If Me.TechAppliedByModel(t) Then
                    If CAFE.Technologies.IsMaterialSubstitution(Me.NRC_ClassIdx, t) Then
                        Dim kw As Double = CAFE.Technologies.GetKWeight(NRC_ClassIdx, t)
                        CWCoeff = (1 - kw)
                        DeltaCW = OldCW - (OldCW * CWCoeff)
                        OldCW = OldCW - DeltaCW
                        tc += CAFE.Technologies.GetCost(Me.NRC_ClassIdx, t) * DeltaCW
                    Else
                        tc += CAFE.Technologies.GetCost(Me.NRC_ClassIdx, t)
                    End If
                End If
            Next
            Return tc
        End Get
        'Get
        '    Dim tc As Double = Me.TechCostVehTechnologies
        '    If IsNothing(CAFE.Engines) = False AndAlso CAFE.Engines.Length > Me.TrueEngineCode Then
        '        tc += CAFE.Engines(TrueEngineCode).TechCost
        '    End If
        '    If IsNothing(CAFE.Transmissions) = False AndAlso CAFE.Transmissions.Length > Me.TrueTransmissionCode Then
        '        tc += CAFE.Transmissions(TrueTransmissionCode).TechCost
        '    End If
        '    If IsNothing(CAFE.Mfgs(Me.MfgIdx).NamePlates) = False AndAlso CAFE.Mfgs(Me.MfgIdx).NamePlates.Length > Me.NamePlateIdx Then
        '        tc += CAFE.Mfgs(Me.MfgIdx).NamePlates(Me.NamePlateIdx).TechCost
        '    End If
        '    Return tc
        'End Get
    End Property
    Public Property ActualFE_FFVs() As Double
        Get
            Return Me._ActualFE_FFVs
        End Get
        Set(ByVal Value As Double)
            Me._ActualFE_FFVs = Value
        End Set
    End Property
    Public Property CodedFuelEconomy() As Double
        Get
            Return Me._FuelEconomy
        End Get
        Set(ByVal Value As Double)
            Me._FuelEconomy = Value
        End Set
    End Property
#End Region

    Sub New()
        nSalesYears = 0
        nMSRPYears = 0
        nPriceYears = 0
        Valid = False
        PredecessorUsed = False

        RegClass = 0
        RegCost = 0
        TechCostVehTechnologies = 0
        Me.TechImprvVehTechnologies = 1
        Utility = 0
        PredInfo.Populated = False

```

```
311         TechUsedStack = New TechnologiesStackClass
312     End Sub
313
314     Sub Copy(ByRef Dst As VehiclesClass)
315         VehiclesClass.Copy(Dst, Me)
316     End Sub
317
318     Public Shared Sub Copy(ByRef Dst As VehiclesClass, ByRef Src As VehiclesClass)
319         Dim i, j As Integer
320
321         Dst.Valid = Src.Valid
322         Dst.ValidOverAll = Src.ValidOverAll
323         Dst.ValidForScenario = Src.ValidForScenario
324         Dst.ValidForMY = Src.ValidForMY
325         Dst.MyIndexNumber = Src.MyIndexNumber
326         Dst.MinYear = Src.MinYear
327         Dst.Number = Src.Number
328         Dst.Manufacturer = Src.Manufacturer
329         Dst.MfgIdx = Src.MfgIdx
330         Dst.Model = Src.Model
331         Dst.NamePlate = Src.NamePlate
332         Dst.NamePlateIdx = Src.NamePlateIdx
333         Dst._FuelEconomy = Src._FuelEconomy
334         Dst.InitialFuelEconomy = Src.InitialFuelEconomy
335         Dst._ActualFE_FFVs = Src._ActualFE_FFVs
336         Dst.EngineCode = Src.EngineCode
337         Dst.TrueEngineCode = Src.TrueEngineCode
338         Dst.TransmissionCode = Src.TransmissionCode
339         Dst.TrueTransmissionCode = Src.TrueTransmissionCode
340         Dst.Origin = Src.Origin
341         Dst.US_Canadian_Content = Src.US_Canadian_Content
342         'Dst.FinalAssemblyCity = Src.FinalAssemblyCity
343         'Dst.FinalAssemblyState = Src.FinalAssemblyState
344         'Dst.FinalAssembly = Src.FinalAssembly
345         'Dst.GeneralNotes = Src.GeneralNotes
346         Dst.Predecessor = Src.Predecessor
347         Dst.PredecessorUsed = Src.PredecessorUsed
348         Dst.PredecessorFromFile = Src.PredecessorFromFile
349         If Src.PredInfo.Populated Then
350             Dst.PredInfo.FE_After = Src.PredInfo.FE_After
351             Dst.PredInfo.FE_Before = Src.PredInfo.FE_Before
352             Dst.PredInfo.MY = Src.PredInfo.MY
353             Dst.PredInfo.Populated = Src.PredInfo.Populated
354             Dst.PredInfo.PredIdx = Src.PredInfo.PredIdx
355             Dst.PredInfo.TechCost = Src.PredInfo.TechCost
356             Dst.PredInfo.TechImprv = Src.PredInfo.TechImprv
357             Dst.PredInfo.Type = Src.PredInfo.Type
358             ReDim Dst.PredInfo.TechUsed(Src.PredInfo.TechUsed.Length - 1)
359             For i = 0 To Src.PredInfo.TechUsed.Length - 1
360                 Dst.PredInfo.TechUsed(i) = Src.PredInfo.TechUsed(i)
361             Next i
362         End If
363         Dst.LastFreshening = Src.LastFreshening
364         Dst.NextFreshening = Src.NextFreshening
365         Dst.LastRedesign = Src.LastRedesign
366         Dst.NextRedesign = Src.NextRedesign
367         Dst.EmploymentHoursPerVehicle = Src.EmploymentHoursPerVehicle
368         'Dst.PlanningNotes = Src.PlanningNotes
369         Dst.nSalesYears = Src.nSalesYears
370         ReDim Dst.Sales(Src.nSalesYears - 1)
371         For i = 0 To Src.nSalesYears - 1
372             Dst.Sales(i) = Src.Sales(i)
373         Next i
374         ReDim Dst.InitialSales(Src.nSalesYears - 1)
375         For i = 0 To Src.nSalesYears - 1
376             Dst.InitialSales(i) = Src.InitialSales(i)
377         Next i
378         ReDim Dst.ModeledBaselineSales(Src.nSalesYears - 1) ' KG:  added 08/06/04
379         For i = 0 To Src.nSalesYears - 1 ' KG:  added 08/06/04
380             Dst.ModeledBaselineSales(i) = Src.ModeledBaselineSales(i) ' KG:  added 08/06/
```

```
04
        Next i ' KG:  added 08/06/04
        'Dst.SalesNotes = Src.SalesNotes
        Dst.nMSRPYears = Src.nMSRPYears
        ReDim Dst.MSRP(Src.nMSRPYears - 1)
        For i = 0 To Src.nMSRPYears - 1
            Dst.MSRP(i) = Src.MSRP(i)
        Next i
        ReDim Dst.InitialMSRP(Src.nMSRPYears - 1)
        For i = 0 To Src.nMSRPYears - 1
            Dst.InitialMSRP(i) = Src.InitialMSRP(i)
        Next i
        'Dst.MSRPNotes = Src.MSRPNotes
        Dst.nPriceYears = Src.nPriceYears
        ReDim Dst.Price(Src.nPriceYears - 1)
        For i = 0 To Src.nPriceYears - 1
            Dst.Price(i) = Src.Price(i)
        Next i
        ReDim Dst.InitialPrice(Src.nPriceYears - 1)
        For i = 0 To Src.nPriceYears - 1
            Dst.InitialPrice(i) = Src.InitialPrice(i)
        Next i
        'Dst.PriceNotes = Src.PriceNotes
        Dst.HybridType = Src.HybridType
        Dst.VoltageOrPressure = Src.VoltageOrPressure
        Dst.EnergyStorageCapacity = Src.EnergyStorageCapacity
        Dst.BatteryType = Src.BatteryType
        Dst.EnergyTransfer = Src.EnergyTransfer
        Dst.BrakingEnergyRecovery = Src.BrakingEnergyRecovery
        Dst.ShareMaximumPower = Src.ShareMaximumPower
        'Dst.HybridizationNotes = Src.HybridizationNotes
        Dst.Style = Src.Style
        Dst.VehClass = Src.VehClass
        Dst.VehStructure = Src.VehStructure
        Dst.Drive = Src.Drive
        Dst.FinalDriveRatio = Src.FinalDriveRatio
        Dst.EngineSpeedOverVehSpeed = Src.EngineSpeedOverVehSpeed
        'Dst.FrontAxleViscosity = Src.FrontAxleViscosity
        'Dst.RearAxleViscosity = Src.RearAxleViscosity
        'Dst.OverAllLength = Src.OverAllLength '[KG:  changed 01/25/05 for footprint-based ↙
 system]
        'Dst.OverAllWidth = Src.OverAllWidth '[KG:  changed 01/25/05 for footprint-based  ↙
system]
        Dst.Wheelbase = Src.Wheelbase '[KG:  changed 01/25/05 for footprint-based system]
        Dst.TrackWidthFront = Src.TrackWidthFront '[KG:  changed 01/25/05 for footprint-  ↙
based system]
        Dst.TrackWidthRear = Src.TrackWidthRear '[KG:  changed 01/25/05 for footprint-    ↙
based system]
        Dst.OverAllArea = Src.OverAllArea
        Dst.OverAllHeight = Src.OverAllHeight
        'Dst.GroundClearance = Src.GroundClearance
        'Dst.FrontAxleClearance = Src.FrontAxleClearance
        'Dst.RearAxleClearance = Src.RearAxleClearance
        'Dst.AngleOfApproach = Src.AngleOfApproach
        'Dst.BreakoverAngle = Src.BreakoverAngle
        'Dst.AngleOfDeparture = Src.AngleOfDeparture
        'Dst.CenterOfGravityHeight = Src.CenterOfGravityHeight
        Dst.CurbWeight = Src.CurbWeight
        Dst.InitialCurbWeight = Src.InitialCurbWeight
        Dst.TestWeight = Src.TestWeight
        Dst.PAUSetting = Src.PAUSetting
        Dst.GVWR = Src.GVWR
        Dst.StandardTowingCapacity = Src.StandardTowingCapacity
        Dst.MaxTowingCapacity = Src.MaxTowingCapacity
        Dst.Payload = Src.Payload
        Dst.MinSeating = Src.MinSeating
        Dst.MaxSeating = Src.MaxSeating
        'Dst.FrontRowSeating = Src.FrontRowSeating
        'Dst.CargoVolBehindFirstRow = Src.CargoVolBehindFirstRow
        'Dst.SecondRowSeating = Src.SecondRowSeating
```

```
446         'Dst.SecondRowFlatCapability = Src.SecondRowFlatCapability
447         'Dst.CargoVolBehindSecondRow = Src.CargoVolBehindSecondRow
448         'Dst.ThirdRowSeating = Src.ThirdRowSeating
449         'Dst.ThirdRowFlatCapability = Src.ThirdRowFlatCapability
450         'Dst.CargoVolBehindThirdRow = Src.CargoVolBehindThirdRow
451         'Dst.EnclosedVolume = Src.EnclosedVolume
452         'Dst.StandardPassengerVolume = Src.StandardPassengerVolume
453         'Dst.MaxPassengerVolume = Src.MaxPassengerVolume
454         'Dst.CargoVolumeIndex = Src.CargoVolumeIndex
455         'Dst.StandardEnclosedCargoVolume = Src.StandardEnclosedCargoVolume
456         'Dst.MaxEnclosedCargoVolume = Src.MaxEnclosedCargoVolume
457         'Dst.OpenBoxAreaLength = Src.OpenBoxAreaLength
458         'Dst.MinOpenBoxAreaWidth = Src.MinOpenBoxAreaWidth
459         'Dst.MaxOpenBoxAreaWidth = Src.MaxOpenBoxAreaWidth
460         Dst.OpenBoxArea = Src.OpenBoxArea
461         Dst.OpenBoxHeight = Src.OpenBoxHeight
462         Dst.FuelCapacity = Src.FuelCapacity
463         'Dst.TireRollingResistance = Src.TireRollingResistance
464         'Dst.FrontalArea = Src.FrontalArea
465         'Dst.RecommendedTirePressure = Src.RecommendedTirePressure
466         'Dst.TireStaticLoadedRadius = Src.TireStaticLoadedRadius
467         'Dst.TireTreadWidth = Src.TireTreadWidth
468         'Dst.TireDeflection = Src.TireDeflection
469         'Dst.AerodynamicDragCoefficient = Src.AerodynamicDragCoefficient
470         'Dst.Percent2DR = Src.Percent2DR
471         'Dst.VehicleNotes = Src.VehicleNotes
472         'Dst.EPA_Class = Src.EPA_Class
473         'Dst.EPA_Certification = Src.EPA_Certification
474         'Dst.LEV_Class = Src.LEV_Class
475         'Dst.RegulatoryNotes = Src.RegulatoryNotes
476
477         ReDim Dst.StageAnalysisFE(Src.StageAnalysisFE.Length - 1)
478         For i = 0 To Src.StageAnalysisFE.Length - 1
479             Dst.StageAnalysisFE(i) = Src.StageAnalysisFE(i)
480         Next i
481         ReDim Dst.StageAnalysisCW(Src.StageAnalysisCW.Length - 1)
482         For i = 0 To Src.StageAnalysisCW.Length - 1
483             Dst.StageAnalysisCW(i) = Src.StageAnalysisFE(i)
484         Next i
485         ReDim Dst.StageAnalysisSales(Src.StageAnalysisSales.Length - 1)
486         For i = 0 To Src.StageAnalysisSales.Length - 1
487             Dst.StageAnalysisSales(i) = Src.StageAnalysisFE(i)
488         Next i
489
490         Dst.LTDFN1 = Src.LTDFN1
491         Dst.LTDFN2 = Src.LTDFN2
492         Dst.LTDFN3 = Src.LTDFN3
493         Dst.LTDFN4 = Src.LTDFN4
494         Dst.LTDFN5 = Src.LTDFN5
495         Dst.HLTINCL1 = Src.HLTINCL1
496         Dst.HLTINCL2 = Src.HLTINCL2
497         Dst.HLTINCL3 = Src.HLTINCL3
498         Dst.HLTINCL4 = Src.HLTINCL4
499         Dst.HLTINCL5 = Src.HLTINCL5
500         'Dst.MOBILE6Class = Src.MOBILE6Class
501         Dst.SafetyClass = Src.SafetyClass
502         Dst.Percent2DR = Src.Percent2DR
503         Dst.MarketSegment_VOLPE = Src.MarketSegment_VOLPE
504         Dst.MarketSegment_AutoNews = Src.MarketSegment_AutoNews
505         'Dst.MarketNest = Src.MarketNest        ' replaced by market segment
506         Dst.TechOverROLL = Src.TechOverROLL
507         Dst.TechOverLDB = Src.TechOverLDB
508         Dst.TechOverFAD = Src.TechOverFAD
509         Dst.TechOverEPS = Src.TechOverEPS
510         Dst.TechOverEAI = Src.TechOverEAI
511         Dst.TechOverAER = Src.TechOverAER
512         Dst.TechOver42V = Src.TechOver42V
513         Dst.TechOverISG = Src.TechOverISG
514         Dst.TechOverMS1 = Src.TechOverMS1
515         Dst.TechOverMS2 = Src.TechOverMS2
```

```
516         Dst.TechOverMS3 = Src.TechOverMS3
517         Dst.TechOverMSX = Src.TechOverMSX
518
519         Dst.NRC_Class = Src.NRC_Class
520         Dst.NRC_ClassIdx = Src.NRC_ClassIdx
521
522         Dst.RegClass = Src.RegClass
523         Dst.RegCost = Src.RegCost
524         Dst.TechCostVehTechnologies = Src.TechCostVehTechnologies
525         Dst.TechImprvVehTechnologies = Src.TechImprvVehTechnologies
526         ReDim Dst.TechCostByMY(Src.TechCostByMY.Length - 1)
527         For i = 0 To Src.TechCostByMY.Length - 1
528             Dst.TechCostByMY(i) = Src.TechCostByMY(i)
529         Next
530         If Not IsNothing(Src.ModeledBaselineSales) Then
531             ReDim Dst.ModeledBaselineSales(Src.ModeledBaselineSales.GetUpperBound(0))
532             For i = 0 To Src.ModeledBaselineSales.GetUpperBound(0)
533                 Dst.ModeledBaselineSales(i) = Src.ModeledBaselineSales(i)
534             Next i
535         End If
536         Dst.Utility = Src.Utility
537         If Not IsNothing(Src.Elasticity) Then
538             ReDim Dst.Elasticity(Src.Elasticity.GetUpperBound(0), Src.Elasticity.
    GetUpperBound(1))
539             For i = 0 To Src.Elasticity.GetUpperBound(0)
540                 For j = 0 To Src.Elasticity.GetUpperBound(1)
541                     Dst.Elasticity(i, j) = Src.Elasticity(i, j)
542                 Next j
543             Next i
544         End If
545         Dst.SalesOverFE = Src.SalesOverFE
546         Dst.TechClass = Src.TechClass
547         If Not IsNothing(Src.TechEnabled) Then
548             ReDim Dst.TechEnabled(Src.TechEnabled.Length - 1)
549             For i = 0 To Src.TechEnabled.Length - 1
550                 Dst.TechEnabled(i) = Src.TechEnabled(i)
551             Next i
552         End If
553         If Not IsNothing(Src.TechApplies) Then
554             ReDim Dst.TechApplies(Src.TechApplies.Length - 1)
555             For i = 0 To Src.TechApplies.Length - 1
556                 Dst.TechApplies(i) = Src.TechApplies(i)
557             Next i
558         End If
559         If Not IsNothing(Src.TechUsed) Then
560             ReDim Dst.TechUsed(Src.TechUsed.Length - 1)
561             For i = 0 To Src.TechUsed.Length - 1
562                 Dst.TechUsed(i) = Src.TechUsed(i)
563             Next i
564         End If
565         If Not IsNothing(Src.TechReserved) Then
566             ReDim Dst.TechReserved(Src.TechReserved.Length - 1)
567             For i = 0 To Src.TechReserved.Length - 1
568                 Dst.TechReserved(i) = Src.TechReserved(i)
569             Next i
570         End If
571         If Not IsNothing(Src.TechAppliedByModel) Then
572             ReDim Dst.TechAppliedByModel(Src.TechAppliedByModel.Length - 1)
573             For i = 0 To Src.TechAppliedByModel.Length - 1
574                 Dst.TechAppliedByModel(i) = Src.TechAppliedByModel(i)
575             Next i
576         End If
577         If Not IsNothing(Src.TechUsedByMY) Then
578             ReDim Dst.TechUsedByMY(Src.TechUsedByMY.Length - 1)
579             For i = 0 To Src.TechUsedByMY.Length - 1
580                 If Not IsNothing(Src.TechUsedByMY(i)) Then
581                     ReDim Dst.TechUsedByMY(i)(Src.TechUsedByMY(i).Length - 1)
582                     For j = 0 To Src.TechUsedByMY(i).Length - 1
583                         Dst.TechUsedByMY(i)(j) = Src.TechUsedByMY(i)(j)
584                     Next j
```

```
585                 End If
586             Next i
587         End If
588         If Not IsNothing(Src.TechIgnored) Then
589             ReDim Dst.TechIgnored(Src.TechIgnored.Length - 1)
590             For i = 0 To Src.TechIgnored.Length - 1
591                 Dst.TechIgnored(i) = Src.TechIgnored(i)
592             Next i
593         End If
594         Dst.TechMS2Enablable = Src.TechMS2Enablable
595         Dst.TechMS3Enablable = Src.TechMS3Enablable
596         Dst.ScaleA5 = Src.ScaleA5
597         Dst.ScaleA6 = Src.ScaleA6
598         Dst.ScaleCVT = Src.ScaleCVT
599     End Sub
600
601     Public Function GetMY() As ArrayList
602         Dim i As Integer
603         Dim MY_Array As ArrayList
604         For i = 0 To Me.nSalesYears - 1
605             MY_Array.Add(Me.Sales(i))
606         Next
607         Return MY_Array
608     End Function
609
610     Public Sub ApplyTechnology(ByVal TechIdx As Integer, ByVal MY As Integer)
611         Dim imy As Integer = CAFE.GMYI(MY)
612         ApplyTechnology_(TechIdx, imy)
613     End Sub
614     Public Sub ApplyTechnology_(ByVal TechIdx As Integer, ByVal iMY As Integer)
615         Me.TechUsed(TechIdx) = True
616         Me.TechAppliedByModel(TechIdx) = True
617         Me.TechUsedByMY(iMY)(TechIdx) = True
618     End Sub
619     Public Sub RemoveTechnology(ByVal TechIdx As Integer, ByVal MY As Integer)
620         Dim imy As Integer = CAFE.GMYI(MY)
621         RemoveTechnology_(TechIdx, imy)
622     End Sub
623     Public Sub RemoveTechnology_(ByVal TechIdx As Integer, ByVal iMY As Integer)
624         Me.TechUsed(TechIdx) = False
625         Me.TechAppliedByModel(TechIdx) = False
626         Me.TechUsedByMY(iMY)(TechIdx) = False
627     End Sub
628
629
630
631
632
633
634
635
636
637
638

639
640
641

642
643
644
645
646
647

648
649
650
```

```
699
700
701



702
703
704
705
706
707
708
709
710
711     ...
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732

733
734
735

736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
```

```
764
765
766
767
768
769
770
771
772

773
774
775
776
777
778
779
780
781
782
783
784
785

786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818

819
820
821
822
823
824
825     '''
826     ' Returns the initial fuel economy (product plan fuel economy) based on the fuel type.
827     '
828     Private Function GetProdPlanCPM() As Double()
829         Return GetCPM(Me.InitialFuelEconomy)
830     End Function
```

```
831
832     Private Function GetProdPlanVMT() As Double()
833         Dim prodPlanVMT(25) As Double
834         Dim i As Integer
835         Dim MilesDriven(,) As Double
836         Dim survRate(,) As Double
837
838         MilesDriven = Me.GetMilesDriven()
839         survRate = Me.GetSurvivalRate()
840
841         ' calculate the product plan VMT
842         For i = 0 To 25
843             prodPlanVMT(i) = Me.InitialSales(i) * survRate(i, 0) * MilesDriven(i, 0) /
    1000
844         Next
845         Return prodPlanVMT
846     End Function
847
848     '''
849     ' Calculates Drive Surplus
850     '
851     Private Sub CalculateDriveSurplus(ByVal Scenario As Integer, ByVal MY As Integer)
852         Dim i As Integer
853         Dim MYIdx As Integer = CAFE.GMYI(MY)
854         Dim baseCPM() As Double
855         Dim altCPM() As Double
856         Dim rebound As Double
857         Dim MilesDriven(,) As Double
858         Dim baseVMT(25) As Double
859         Dim prodPlanDriveSurplus(25) As Double
860         'Dim prodPlanCPM() As Double
861         'Dim prodPlanVMT() As Double
862
863         ReDim driveSurplus(25)
864         ReDim discDriveSurplus(25)
865         ReDim initialDriveSurplus(25)
866
867         MilesDriven = Me.GetMilesDriven()
868         'prodPlanVMT = Me.GetProdPlanVMT()
869
870         ' calculate the base VMT
871         For i = 0 To 25
872             baseVMT(i) = Me.Fleet(i) * MilesDriven(i, 0) / 1000
873         Next i
874
875         'prodPlanCPM = Me.GetProdPlanCPM()
876         rebound = Me.GetReboundEffect
877         altCPM = Me.GetAltCPM()
878
879         If Scenario = 0 Then
880             ' this is the baseline scenario
881             baseCPM = altCPM
882         Else
883             ' this is the alternative scenario
884             baseCPM = Me.GetBaseCPM(MY)
885         End If
886
887         ' calculate the drive surplus = 0.5 * (baseVMT * ((altCPM/baseCPM) - 1) * rebound)
     * (baseCPM - altCPM)
888         For i = 0 To 25
889             driveSurplus(i) = -0.5 * (baseVMT(i) * ((altCPM(i) / baseCPM(i)) - 1) *
    rebound) * (baseCPM(i) - altCPM(i))
890             discDriveSurplus(i) = driveSurplus(i) * ((1 + CAFE.Parameters.discountRate) ^
    -i)
891             'initialDriveSurplus(i) = 0.5 * (prodPlanVMT(i) * ((altCPM(MY) / prodPlanCPM
    (MY)) - 1) * rebound) * (prodPlanCPM(MY) - altCPM(MY))
892         Next
893
894     End Sub
895
```

```
896     '''
897     ' calculates range surplus
898     '
899     Private Sub CalculateRefuelSurplus(ByVal Scenario As Integer, ByVal MY As Integer)
900         Dim i As Integer
901         Dim MYIdx As Integer = CAFE.GMYI(MY)
902         Dim baseCPM() As Double
903         Dim altCPM() As Double
904         Dim rebound As Double
905         Dim fuelTank As Double
906         Dim MilesDriven(,) As Double
907         Dim prodPlanRefuelSurplus(25) As Double
908         'Dim prodplanCPM() As Double
909         'Dim prodPlanVMT() As Double
910
911
912         ReDim refuelSurplus(25)
913         ReDim discRefuelSurplus(25)
914         ReDim initialRefuelSurplus(25)
915
916         MilesDriven = Me.GetMilesDriven()
917         altCPM = Me.GetAltCPM()
918
919         If Scenario = 0 Then
920             ' this is the baseline scenario
921             baseCPM = altCPM
922         Else
923             ' this is the alternative scenario
924             baseCPM = Me.GetBaseCPM(MY)
925         End If
926
927         'prodplanCPM = Me.GetProdPlanCPM()
928         rebound = Me.GetReboundEffect()
929         fuelTank = Me.FuelCapacity
930         'prodPlanVMT = GetProdPlanVMT()
931
932         Dim BaseFE As Double        ' the baseline fuel economy, from the stack
933         Dim VehsStack As VehiclesStackClass
934
935         If Scenario = 0 Then
936             ' set BaseFE to be the current fuel economy since there is no baseline values yet
937             BaseFE = Me.FuelEconomy
938         Else
939             ' get the baseline fuel economy from the stack
940             ' the baseline stack is at the 0-th index
941             ' jvs 051005 -- Remove references to stacks for Monte Carlo analysis
942             'VehsStack = CAFE.VehiclesStack(0)(Me.MyIndexNumber)
943
944             '' scan the baseline stack for the current model, to get the FE
945             'For i = 0 To VehsStack.Size - 1
946             '    If VehsStack.Read(i).MY = MY Then
947             '        ' we found a match
948             '        BaseFE = CType(VehsStack.Read(i), VehicleStackClass).FuelEconomyFinal
949             '        Exit For
950             '    End If
951             'Next i
952             ' jvs 051005 -- replace stack reference with this:
953             BaseFE = CAFE.BaseLineInfo(CAFE.GetMYearIndex(MY)).FinalFuelEconomy(Me.MyIndexNumber)
954         End If
955
956         ' calculate refuel surplus = ((miles / (tank * (1-gap) * ActFEBase)) -
957         ' (miles * (1 + ((AltCPM - BaseCPM) - 1) * Rebound)) / (tank * (1-gap) * ActFEAlt))
958         ' * (5/60) * (travel time value/veh)/1000
959
960         For i = 0 To 25
961             refuelSurplus(i) = -((MilesDriven(i, 0) / (fuelTank * (1 - CAFE.Parameters.gapTestOnRoad) * BaseFE)) - _
```

```
962                                 (MilesDriven(i, 0) * (1 + ((altCPM(i) / baseCPM(i)) - 1) * rebound) / (fuelTank * (1 - CAFE.Parameters.gapTestOnRoad) * Me.FuelEconomy))) * _
963                                 Me.Fleet(i) * ((5 / 60) * CAFE.Parameters.travelValueVehicle / 1000)
964             discRefuelSurplus(i) = refuelSurplus(i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
965             'initialRefuelSurplus(i) = ((MilesDriven(i, 0) / (fuelTank * (1 - CAFE.Parameters.gapTestOnRoad) * Me.InitialFuelEconomy)) - ((MilesDriven(i, 0) * (1 + ((altCPM(MY) - prodplanCPM(MY)) - 1) * rebound)) / (fuelTank * (1 - CAFE.Parameters.gapTestOnRoad) * Me.FuelEconomy))) * ((5 / 60) * CAFE.Parameters.travelValueVehicle / 1000)
966         Next
967
968     End Sub
969
970     '''
971     ' Pre processes this vehicle.  Sets the regulatory class for this vehicle and allocates used
972     ' technologies.
973     '
974     Public Sub PreProcess(ByVal Scenario As Integer, ByVal MY As Integer)
975         Dim nMY As Integer = CAFE.ModelYearsSelected.Length
976
977         If Me.IsValid Then
978             Dim nTechnologies As Integer = TechnologiesClass.nTotalTechnologies
979
980             ' ReDim(ModeledBaselineSales(nMY - 1))
981             ReDim TechEnabled(nTechnologies - 1)
982             If IsNothing(TechUsedByMY) Then
983                 ReDim TechUsedByMY(nMY - 1)
984             End If
985             ReDim TechUsedByMY(CAFE.GMYI(MY))(nTechnologies - 1)
986             ReDim TechIgnored(nTechnologies - 1)
987
988             TechCostByMY(CAFE.GMYI(MY)) = 0                         ' clear tech cost for this MY
989
990
991             ' get the reg class for this vehicle based on the given MY
992             Dim RegClass As Integer = CAFE.Standards.GetRegClass(Scenario, MY, Me.MyIndexNumber)
993
994             ' assign the reg class for this vehicle
995             CAFE.Vehicles(Me.MyIndexNumber).RegClass = RegClass
996
997
998             Me.RegCost = 0
999
1000            ' assign the mobile6 class for this vehicle
1001            Me.MOBILE6Class = Me.AssignMobile6Class()
1002        Else
1003            ' TO DO:
1004            ' If the only reason this veh is invalid is MY reasons, then log it.
1005        End If
1006    End Sub
1007
1008    ' PredecessorsAndLegacies
1009    ' Look for predecessor and legacy technology application which should go on this veh.
1010    '
1011    Public Sub PredecessorsAndLegacies(ByVal Scenario As Integer, ByVal MY As Integer)
1012        Dim Pred As Integer = Me.Predecessor
1013        Dim PredImprv As Double = 1
1014        Dim PredCost As Double = 0
1015        Dim PredEng As Integer
1016        Dim PredTran As Integer
1017        Dim eidx As Integer = Me.TrueEngineCode
1018        Dim tidx As Integer = Me.TrueTransmissionCode
1019        Dim npidx As Integer = Me.NamePlateIdx
1020        Dim CWCoeff As Double
1021        Dim DeltaCW As Double
```

```
1022        Dim OldCW As Double = Me.CurbWeight
1023        Dim WeightChange As Boolean
1024
1025        PredInfo.Populated = False
1026        If Scenario = 3 And MY = 2010 And Me.MyIndexNumber = 674 Then
1027            Debug.Write("")
1028        End If
1029        If Me.IsValid Then
1030            If Not IsNothing(Me.Predecessor) And Me.Predecessor >= 0 And PredecessorUsed = ↙
     False Then
1031
1032                PredEng = CAFE.Vehicles(Pred).TrueEngineCode
1033                PredTran = CAFE.Vehicles(Pred).TrueTransmissionCode
1034
1035                ReDim Me.PredInfo.TechUsed(TechnologiesClass.nTotalTechnologies)
1036                Dim ATechFound As Boolean = False
1037                For t As Integer = 0 To Me.TechAppliedByModel.Length - 1
1038                    Dim tt As String = CAFE.Technologies.VehicleClasses(0).Technology(t). ↙
    TechType.ToUpper
1039                    If tt = "ENGMOD" Then
1040                        PredInfo.TechUsed(t) = (CAFE.Engines(eidx).TechUsed(t) And Not Me. ↙
    TechUsed(t) And Me.TechEnabled(t))
1041                    ElseIf tt = "TRMOD" Then
1042                        PredInfo.TechUsed(t) = (CAFE.Transmissions(tidx).TechUsed(t) And   ↙
    Not Me.TechUsed(t) And Me.TechEnabled(t))
1043                    ElseIf tt = "AERO" Then
1044                        PredInfo.TechUsed(t) = (CAFE.Mfgs(MfgIdx).NamePlates(npidx).       ↙
    TechUsed(t) And Not Me.TechUsed(t) And Me.TechEnabled(t))
1045                    Else
1046                        PredInfo.TechUsed(t) = (CAFE.Vehicles(Pred).TechUsed(t) And Not    ↙
    TechUsed(t) And Me.TechEnabled(t))
1047                    End If
1048                    If PredInfo.TechUsed(t) Then
1049                        TechUsed(t) = True
1050                        TechAppliedByModel(t) = True
1051                        PredImprv = PredImprv * (1 - CAFE.Technologies.GetImprv(Me.        ↙
    NRC_ClassIdx, t))
1052                        PredCost = PredCost + CAFE.Technologies.GetCost(Me.NRC_ClassIdx,   ↙
    t)
1053                        If CAFE.Technologies.IsMaterialSubstitution(Me.NRC_ClassIdx, t)    ↙
    Then
1054                            If WeightChange = False Then PredInfo.OrigCW = OldCW
1055                            Dim kw As Double = CAFE.Technologies.GetKWeight(NRC_ClassIdx, ↙
    t)
1056                            CWCoeff = (1 - kw)
1057                            DeltaCW = OldCW - (OldCW * CWCoeff)
1058                            CurbWeight = CurbWeight - DeltaCW
1059                            PredInfo.NewCW = CurbWeight
1060                            OldCW = CurbWeight
1061                            WeightChange = True
1062                        End If
1063                    End If
1064                    ATechFound = ATechFound Or PredInfo.TechUsed(t)
1065                Next t
1066
1067                If Me.HybridType = String.Empty And CAFE.Vehicles(Pred).HybridType <>      ↙
    String.Empty Then
1068                    Me.HybridType = CAFE.Vehicles(Pred).HybridType
1069                End If
1070                If Me.MOBILE6Class = String.Empty And CAFE.Vehicles(Pred).MOBILE6Class <>  ↙
    String.Empty Then
1071                    Me.MOBILE6Class = CAFE.Vehicles(Pred).MOBILE6Class
1072                End If
1073
1074                If Me.CurbWeight = 0 Then
1075                    Me.CurbWeight = CAFE.Vehicles(Pred).CurbWeight
1076                End If
1077
1078                ' Record this pred info for posterity
1079                PredInfo.MY = MY
```

```
1080                PredInfo.FE_Before = Me.FuelEconomy
1081                PredInfo.TechCost = PredCost
1082                PredInfo.PredIdx = Pred
1083                PredInfo.Type = "P"
1084
1085
1086                ' Adjust the FE to any improvements made in Engine, Transmission or Nameplate
1087                'MyEImprv = CAFE.Engines(Me.TrueEngineCode).TechImprv
1088                'MyTImprv = CAFE.Transmissions(Me.TrueTransmissionCode).TechImprv
1089                'MyNImprv = CAFE.Mfgs(Me.MfgIdx).NamePlates(Me.NamePlateIdx).TechImprv
1090                'Me.FuelEconomy = Me.FuelEconomy / (MyVImprv * MyEImprv * MyTImprv * MyNImprv)
1091                Me.FuelEconomy = Me.FuelEconomy / PredImprv
1092
1093                ' finish the paperwork
1094                'PredInfo.TechImprv = (MyVImprv * MyEImprv * MyTImprv * MyNImprv)
1095                PredInfo.TechImprv = PredImprv
1096                PredInfo.FE_After = Me.FuelEconomy
1097                PredInfo.Populated = (PredInfo.TechCost > 0) And ATechFound
1098                PredecessorUsed = True
1099            Else
1100                ' No predecessor to borrow from
1101
1102                ' Check if this vehicle's tech usage is in sync with the components.
1103                ' THis could happen if an engine is in use in a previous year on a different veh
1104                ' and that engine gets tech applied. Then this veh comes along, starting up in
1105                ' this MY.  We need to get its' tech info
1106                Dim ATechFound As Boolean = False
1107                Dim AddedTechCost As Double = 0
1108                Dim eimprv As Double = 1
1109                Dim timprv As Double = 1
1110                Dim nimprv As Double = 1
1111                ReDim Me.PredInfo.TechUsed(TechnologiesClass.nTotalTechnologies)
1112                Dim tstate As Boolean
1113
1114                For t As Integer = 0 To Me.TechAppliedByModel.Length - 1
1115                    Dim tt As String = CAFE.Technologies.VehicleClasses(0).Technology(t).TechType.ToUpper
1116                    If tt = "ENGMOD" Then
1117                        tstate = (CAFE.Engines(eidx).TechUsed(t) And Not Me.TechUsed(t))
1118                        If tstate And Me.TechEnabled(t) Then
1119                            PredInfo.TechUsed(t) = tstate
1120                            PredImprv = CAFE.Technologies.GetImprv(Me.NRC_ClassIdx, t)
1121                            eimprv = eimprv * (1 - PredImprv)
1122                        End If
1123                    ElseIf tt = "TRMOD" Then
1124                        tstate = (CAFE.Transmissions(tidx).TechUsed(t) And Not Me.TechUsed(t))
1125                        If tstate And Me.TechEnabled(t) Then
1126                            PredInfo.TechUsed(t) = tstate
1127                            PredImprv = CAFE.Technologies.GetImprv(Me.NRC_ClassIdx, t)
1128                            timprv = timprv * (1 - PredImprv)
1129                        End If
1130                    ElseIf tt = "AERO" Then
1131                        tstate = (CAFE.Mfgs(MfgIdx).NamePlates(npidx).TechUsed(t) And Not Me.TechUsed(t))
1132                        If tstate And Me.TechEnabled(t) Then
1133                            PredInfo.TechUsed(t) = tstate
1134                            PredImprv = CAFE.Technologies.GetImprv(Me.NRC_ClassIdx, t)
1135                            nimprv = nimprv * (1 - PredImprv)
1136                        End If
1137                    End If
1138                    If PredInfo.TechUsed(t) Then
1139                        If CAFE.Technologies.IsMaterialSubstitution(Me.NRC_ClassIdx, t) Then
1140                            If WeightChange = False Then PredInfo.OrigCW = OldCW
1141                            Dim kw As Double = CAFE.Technologies.GetKWeight(NRC_ClassIdx,
```

```
t)
                            CWCoeff = (1 - kw)
                            DeltaCW = OldCW - OldCW * CWCoeff
                            CurbWeight = CurbWeight - DeltaCW
                            PredInfo.NewCW = CurbWeight
                            OldCW = CurbWeight
                            WeightChange = True
                        End If
                        ATechFound = ATechFound Or PredInfo.TechUsed(t)
                        AddedTechCost += CAFE.Technologies.GetCost(Me.NRC_ClassIdx, t)
                        Me.TechUsed(t) = True
                        Me.TechAppliedByModel(t) = True
                    End If
                Next t
                If atechfound Then
                    ' We've had to update this vehicle from at least one component.
                    ' finalize and record changes
                    PredImprv = eimprv * timprv * nimprv
                    PredInfo.Populated = True
                    PredInfo.Type = "L"
                    PredInfo.FE_Before = Me.FuelEconomy
                    PredInfo.PredIdx = 0
                    PredInfo.TechCost = AddedTechCost
                    PredInfo.TechImprv = PredImprv
                    PredInfo.MY = MY

                    Me.FuelEconomy = Me.FuelEconomy / PredImprv
                    PredInfo.FE_After = Me.FuelEconomy
                End If
            End If
        End If
    End Sub

    '''
    ' Post processes this vehicle.  Performs calculations necessary for the effects model.
    '
    Public Sub PostProcess(ByVal Scenario As Integer, ByVal MY As Integer)
        ' if this vehicle is not valid, do not post process it
        If Not Me.IsValid Then
            Exit Sub
        End If

        ' perform these post-processing tasks only if the user requested to run the effects model
        If CAFE.OtherSettings.RunEffectsModel Then
            ' calculate the fleet for this vehicle
            Me.CalculateFleet(MY)
            ' calculate the VMT for this vehicle
            Me.CalculateVMT(Scenario, MY)
            ' calculate the fatalities for this vehicle, only if RunSafetyCalcs is true
            If CAFE.SafetyIncluded Then
                Me.CalculateFatals(MY)
            End If
            ' calculate the gallons for this vehicle
            Me.CalculateGallons()
            ' calculate the pre-tax fuel expenditures
            Me.CalculatePreTaxFuel()
            ' calculates the fuel tax revenues
            Me.CalculateFuelTax()
            ' calculate drive surplus
            Me.CalculateDriveSurplus(Scenario, MY)
            ' calculate refuel surplus
            Me.CalculateRefuelSurplus(Scenario, MY)
        End If
    End Sub

    '''
    ' Set NRC Class info
    ' Call this early in preprocessing (while still MasterVehicles, we presume) to set the NRC class
```

```
1209    '
1210    Public Sub SetNRCClass()
1211        CAFE.Technologies.SetNRC_Class(Me, CAFE.MasterEngines(Me.TrueEngineCode))
1212    End Sub
1213
1214    '''
1215    ' Determines and returns the Mobile6 Class for this vehicle.
1216    '
1217    Public Function AssignMobile6Class() As String
1218        Dim M6Class As String
1219        Dim FuelType As String = CAFE.Engines(Me.TrueEngineCode).Fuel
1220        Dim uStyle As String = UCase(Me.Style)
1221
1222        If uStyle = "CONVERTIBLE" OrElse uStyle = "SEDAN" OrElse uStyle = "WAGON" Then
1223            ' cars
1224            If UCase(FuelType) = "D" Then
1225                ' diesel
1226                M6Class = "LDDV"
1227            Else 'If UCase(FuelType) = "G" Then
1228                ' gas
1229                M6Class = "LDGV"
1230                'Else
1231                '    M6Class = "UNKNOWN"
1232
1233                ' throw an exception for debugging purposes...
1234                dprint("Unhandled fuel type in cars.")
1235            End If
1236            ' / end cars
1237        ElseIf uStyle = "UTILITY" OrElse uStyle = "PICKUP" OrElse uStyle = "VAN" OrElse _
1238               uStyle = "OTHER" OrElse uStyle = "SPORTS UTILITY" Then
1239            '
1240            ' trucks
1241            If UCase(FuelType) = "D" Then
1242                ' diesel
1243                If GVWR <= 6000 Then
1244                    M6Class = "LDDT1"
1245                ElseIf GVWR >= 6001 And GVWR <= 8500 Then
1246                    M6Class = "LDDT2"
1247                ElseIf GVWR >= 8501 And GVWR <= 10000 Then
1248                    M6Class = "HDDV2b"
1249                Else
1250                    M6Class = "UNKNOWN"
1251
1252                    ' throw an exception for debugging purposes...
1253                    dprint("Unhandled GVWR.")
1254                End If
1255                ' / end diesel
1256            Else 'If UCase(FuelType) = "G" Then
1257                ' gas
1258                If GVWR <= 6000 Then
1259                    M6Class = "LDGT1"
1260                ElseIf GVWR >= 6001 And GVWR <= 8500 Then
1261                    M6Class = "LDGT2"
1262                ElseIf GVWR >= 8501 And GVWR <= 10000 Then
1263                    M6Class = "HDGV2b"
1264                Else
1265                    M6Class = "UNKNOWN"
1266
1267                    ' throw an exception for debugging purposes...
1268                    dprint("Unhandled GVWR.")
1269                End If
1270                ' / end gas
1271                'Else
1272                '    M6Class = "UNKNOWN"
1273
1274                ' throw an exception for debugging purposes...
1275                'dprint("Unhandled fuel type in trucks.")
1276            End If
1277            ' / end trucks
1278        Else
```

```
1279            M6Class = "UNKNOWN"
1280
1281            ' throw an exception for debugging purposes...
1282            dprint("Unhandled vehicle style.")
1283        End If
1284
1285        Return M6Class
1286    End Function
1287
1288    '''
1289    ' Calculates the fleet (sales * survival) for this vehicle
1290    ' for 25 years using the sales from the specified MY.
1291    '
1292    Private Sub CalculateFleet(ByVal MY As Integer)
1293        Dim i As Integer
1294        Dim MYIdx As Integer = CAFE.GMYI(MY)
1295        Dim SurvRate(,) As Double
1296
1297        ReDim Me.Fleet(25)
1298        SurvRate = GetSurvivalRate()
1299
1300        For i = 0 To 25
1301            Me.Fleet(i) = Me.Sales(MYIdx) * SurvRate(i, 0)
1302        Next i
1303    End Sub
1304
1305    '''
1306    ' Calculates the VMT for this vehicle for 25 years using
1307    ' fleet and estimated miles driven.
1308    '
1309    Private Sub CalculateVMT(ByVal Scenario As Integer, ByVal MY As Integer)
1310        Dim i As Integer
1311        Dim MYIdx As Integer = CAFE.GMYI(MY)
1312        Dim MilesDriven(,) As Double
1313        Dim TmpVMT(25) As Double
1314        Dim BaseCPM() As Double
1315        Dim ReboundEffect As Double
1316
1317        MilesDriven = GetMilesDriven()
1318
1319        ' calculate the temporary VMT
1320        For i = 0 To 25
1321            TmpVMT(i) = Me.Fleet(i) * MilesDriven(i, 0) / 1000
1322        Next i
1323
1324        If Scenario = 0 Then
1325            ' this is the baseline scenario
1326            ' calculate VMT as: Fleet * Miles / 1000
1327            Me.VMT = TmpVMT
1328            ' save the baseline CPM as well
1329            Me.CPM = Me.GetAltCPM()
1330        Else
1331            ' this is the alternative scenario
1332            ' calculate VMT as: Fleet * Miles / 1000 + ((Fleet * Miles / 1000) * (altCPM / ↙
     baseCPM - 1) * rebound)
1333            BaseCPM = Me.GetBaseCPM(MY)
1334            Me.CPM = Me.GetAltCPM()              ' Me.CPM is actually the alternative CPM
1335            ReboundEffect = Me.GetReboundEffect()
1336
1337            ReDim Me.VMT(25)
1338            For i = 0 To 25
1339                Me.VMT(i) = TmpVMT(i) + (TmpVMT(i) * ((Me.CPM(i) / BaseCPM(i)) - 1) *      ↙
    ReboundEffect)
1340            Next i
1341        End If
1342    End Sub
1343
1344    '''
1345    ' Calculates the gallons for this vehicle for 25 years using
1346    ' VMT values and fuel economy.
```

```
1347    '
1348    Private Sub CalculateGallons()
1349        Dim i As Integer
1350
1351        ReDim Me.Gallons(25)
1352        For i = 0 To 25
1353            Me.Gallons(i) = Me.VMT(i) / (Me.FuelEconomy * (1 - CAFE.Parameters.
        gapTestOnRoad))
1354        Next i
1355    End Sub
1356
1357    '''
1358    ' Calculates the pre-tax fuel expenditures
1359    '
1360    Private Sub CalculatePreTaxFuel()
1361        Dim i As Integer
1362        Dim j As Integer
1363        Dim FuelType As String = CAFE.Engines(Me.TrueEngineCode).Fuel
1364        Dim MYIdx As Integer = CAFE.GMYI(MinYear)
1365
1366        ReDim Me.preTaxFuel(25)
1367        ReDim Me.discPreTaxFuel(25)
1368
1369        For i = 0 To 25
1370            If UCase(FuelType) = "D" Then
1371                'diesel
1372                preTaxFuel(i) = Me.Gallons(i) * CAFE.Parameters.fuelPretaxDies(Math.Min
        (MYIdx + i, 25))
1373                discPreTaxFuel(i) = preTaxFuel(i) * ((1 + CAFE.Parameters.discountRate) ^
        -i)
1374            Else
1375                ' gas
1376                preTaxFuel(i) = Me.Gallons(i) * CAFE.Parameters.fuelPretaxGas(Math.Min
        (MYIdx + i, 25))
1377                discPreTaxFuel(i) = preTaxFuel(i) * ((1 + CAFE.Parameters.discountRate) ^
        -i)
1378            End If
1379        Next
1380
1381        'For i = 0 To 25
1382        '    For j = CAFE.GMYI(MinYear) To 25
1383        '        If UCase(FuelType) = "D" Then
1384        '            'diesel
1385        '            preTaxFuel(i) = Me.Gallons(i) * CAFE.parameters.fuelPretaxDies(j)
1386        '            discPreTaxFuel(i) = preTaxFuel(i) * ((1 + CAFE.parameters.
        discountRate) ^ -i)
1387        '        Else
1388        '            ' gas
1389        '            preTaxFuel(i) = Me.Gallons(i) * CAFE.parameters.fuelPretaxGas(j)
1390        '            discPreTaxFuel(i) = preTaxFuel(i) * ((1 + CAFE.parameters.
        discountRate) ^ -i)
1391        '        End If
1392        '    Next
1393        'Next
1394
1395    End Sub
1396
1397    '''
1398    ' Calculates the fuel tax revenues
1399    '
1400    Private Sub CalculateFuelTax()
1401        Dim i As Integer
1402        Dim j As Integer
1403        Dim FuelType As String = CAFE.Engines(Me.TrueEngineCode).Fuel
1404        Dim MYIdx As Integer = CAFE.GMYI(MinYear)
1405
1406        ReDim Me.fuelTax(25)
1407        ReDim Me.discFuelTax(25)
1408
1409        For i = 0 To 25
```

```
1410            If UCase(FuelType) = "D" Then
1411                'diesel
1412                fuelTax(i) = Me.Gallons(i) * CAFE.Parameters.fuelTaxDiesTot(Math.Min(MYIdx ↙
     + i, 25))
1413                discFuelTax(i) = fuelTax(i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
1414            Else
1415                ' gas
1416                fuelTax(i) = Me.Gallons(i) * CAFE.Parameters.fuelTaxGasTot(Math.Min(MYIdx ↙
    + i, 25))
1417                discFuelTax(i) = fuelTax(i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
1418            End If
1419        Next
1420
1421        'For i = 0 To 25
1422        '    For j = CAFE.GMYI(MinYear) To 25
1423        '        If UCase(FuelType) = "D" Then
1424        '            'diesel
1425        '            fuelTax(i) = Me.Gallons(i) * CAFE.parameters.fuelTaxDiesTot(j)
1426        '            discFuelTax(i) = fuelTax(i) * ((1 + CAFE.parameters.discountRate) ^ - ↙
    i)
1427        '        Else
1428        '            ' gas
1429        '            fuelTax(i) = Me.Gallons(i) * CAFE.parameters.fuelTaxGasTot(j)
1430        '            discFuelTax(i) = fuelTax(i) * ((1 + CAFE.parameters.discountRate) ^ - ↙
    i)
1431        '        End If
1432        '    Next
1433        'Next
1434    End Sub
1435
1436    '''
1437    ' Returns the survival rate for this vehicle, based on its
1438    ' mobile6 class.
1439    '
1440    Private Function GetSurvivalRate() As Double(,)
1441        Dim SurvRate(,) As Double
1442
1443        Select Case UCase(Me.MOBILE6Class)
1444            Case "LDDV"
1445                ' 1
1446                SurvRate = CAFE.Parameters.survivalLDDV
1447            Case "LDGV"
1448                ' 2
1449                SurvRate = CAFE.Parameters.survivalLDGV
1450            Case "LDDT1"
1451                ' 3
1452                SurvRate = CAFE.Parameters.survivalLDDV1
1453            Case "LDGT1"
1454                ' 4
1455                SurvRate = CAFE.Parameters.survivalLDGT1
1456            Case "LDDT2"
1457                ' 5
1458                SurvRate = CAFE.Parameters.survivalLDDV2
1459            Case "LDGT2"
1460                ' 6
1461                SurvRate = CAFE.Parameters.survivalLDGT2
1462            Case "HDDV2B"
1463                ' 7
1464                SurvRate = CAFE.Parameters.survivalHDDV2b
1465            Case "HDGV2B"
1466                ' 8
1467                SurvRate = CAFE.Parameters.survivalHDGV2b
1468            Case Else
1469                ' error case
1470                ReDim SurvRate(25, 0)
1471        End Select
1472
1473        Return SurvRate
1474    End Function
1475
```

```
1476        '''
1477        ' Returns the estimated miles driven for this vehicle, based
1478        ' on its market segment.
1479        '
1480        Private Function GetMilesDriven() As Double(,)
1481            Dim MilesDriven(,) As Double
1482
1483            Select Case Me.MarketSegment_AutoNews
1484                Case 1 To 12
1485                    ' auto
1486                    MilesDriven = CAFE.Parameters.milesAuto
1487                Case 13, 14
1488                    ' pickup
1489                    MilesDriven = CAFE.Parameters.milesPickup
1490                Case 15, 16
1491                    ' van
1492                    MilesDriven = CAFE.Parameters.milesVan
1493                Case 17 To 23
1494                    ' suv
1495                    MilesDriven = CAFE.Parameters.milesSuv
1496                Case Else
1497                    ' default case
1498                    MilesDriven = CAFE.Parameters.milesLTDefault
1499            End Select
1500
1501            Return MilesDriven
1502        End Function
1503
1504        '''
1505        ' Returns the baseCPM based on the fuel type.
1506        '
1507        Private Function GetBaseCPM(ByVal MY As Integer) As Double()
1508            Dim i As Integer
1509            Dim BaseFE As Double        ' the baseline fuel economy, from the stack
1510            ' commented out to eliminate stack use in Monte Carlo mode  jvs  8 apr 05
1511            'Dim VehsStack As VehiclesStackClass
1512
1513            '' get the baseline fuel economy from the stack
1514            '' the baseline stack is at the 0-th index
1515            'VehsStack = CAFE.VehiclesStack(0)(Me.MyIndexNumber)
1516
1517            '' scan the baseline stack for the current model, to get the FE
1518            'For i = 0 To VehsStack.Size - 1
1519            '    If VehsStack.Read(i).MY = MY Then
1520            '        ' we found a match
1521            '        BaseFE = CType(VehsStack.Read(i), VehicleStackClass).FuelEconomyFinal
1522            '        Exit For
1523            '    End If
1524            'Next i
1525            i = CAFE.GetMYearIndex(MY)
1526            BaseFE = CAFE.BaseLineInfo(i).FinalFuelEconomy(Me.MyIndexNumber)
1527
1528            Return GetCPM(BaseFE)
1529        End Function
1530
1531        '''
1532        ' Returns the altCPM based on the fuel type.
1533        '
1534        Private Function GetAltCPM() As Double()
1535            Return GetCPM(Me.FuelEconomy)
1536        End Function
1537
1538        '''
1539        ' Returns the CPM based on the fuel type with the specified fuel economy.
1540        '
1541        Private Function GetCPM(ByVal FuelEconomy As Double) As Double()
1542            Dim i As Integer
1543            Dim CPM() As Double
1544            Dim FuelPrice() As Double
1545            Dim FuelType As String = UCase(CAFE.Engines(Me.TrueEngineCode).Fuel)
```

```
1546            Dim MYIdx As Integer = CAFE.GMYI(MinYear)
1547
1548            If FuelType = "D" Then
1549                ' diesel
1550                FuelPrice = CAFE.Parameters.fuelPriceDiesel
1551            Else
1552                ' gas
1553                FuelPrice = CAFE.Parameters.fuelPriceGas
1554            End If
1555
1556            ReDim CPM(25)
1557            For i = 0 To 25
1558                CPM(i) = FuelPrice(Math.Min(MYIdx + i, 25)) / FuelEconomy * (1 - CAFE.
        Parameters.gapTestOnRoad)
1559            Next i
1560
1561            Return CPM
1562        End Function
1563
1564        '''
1565        ' Returns the rebound effect.
1566        '
1567        Private Function GetReboundEffect() As Double
1568            Return CAFE.Parameters.reboundEffect
1569        End Function
1570
1571
1572    #Region " Validate methods "
1573
1574        Private Function ValidateString(ByVal str As String) As Boolean
1575            ' strings cannot be nothing, empty, or blank
1576            If IsNothing(str) OrElse str = String.Empty OrElse str = "" Then
1577                Return False
1578            Else
1579                Return True
1580            End If
1581        End Function
1582
1583        Private Function ValidateInteger(ByVal int As Integer) As Boolean
1584            ' integers cannot be nothing or 0
1585            If IsNothing(int) OrElse int = 0 Then
1586                Return False
1587            Else
1588                Return True
1589            End If
1590        End Function
1591
1592        Private Function ValidateDouble(ByVal dbl As Double) As Boolean
1593            ' doubles cannot be nothing or 0.0
1594            If IsNothing(dbl) OrElse dbl = 0.0 Then
1595                Return False
1596            Else
1597                Return True
1598            End If
1599        End Function
1600
1601        Private Function ValidateSales() As Boolean
1602            Return ValidateSalesOrMSRP(Me.Sales, Me.nSalesYears)
1603        End Function
1604
1605        Private Function ValidateMSRP() As Boolean
1606            Return ValidateSalesOrMSRP(Me.MSRP, Me.nMSRPYears)
1607        End Function
1608
1609        Private Function ValidateSalesOrMSRP(ByVal arr() As Double, ByVal length As Integer)
        As Boolean
1610            Dim valid As Boolean = False
1611
1612            If IsNothing(arr) Then
1613                Return False
```

```
1614            Else
1615                Dim i As Integer
1616
1617                For i = 0 To length - 1
1618                    If arr(i) > 0 Then
1619                        valid = True
1620                    End If
1621                Next i
1622            End If
1623
1624            ' all checks passed
1625            Return valid
1626        End Function
1627
1628        Public Function ValidateEngineAndTransmission() As Boolean
1629            If Not CAFE.MasterEngines(Me.TrueEngineCode).Valid Then
1630                ' Engine isn't valid... set this vehicle to be invalid
1631                Me.ValidOverAll = False
1632                Me.ReasonForBeingInvalid += "Invalid engine;"
1633            End If
1634
1635            If CAFE.MasterTransmissions(Me.TrueTransmissionCode).Valid = False Then
1636                ' Transmission isn't valid... set vehicle to be invalid
1637                Me.ValidOverAll = False
1638                Me.ReasonForBeingInvalid += "Invalid transmission;"
1639            End If
1640
1641            Return Me.ValidOverAll
1642        End Function
1643
1644        '''
1645        ' IsValid property
1646        ' Gathers the current valid settings (OverAllValid, ScenarioValid, MYValid)
1647        ' and returns the concensus
1648        ' (ok, ok, it's not a concensus, they must all be true for it to be valid)
1649        '
1650        Public ReadOnly Property IsValid() As Boolean
1651            Get
1652                Return ValidOverAll And ValidForScenario And ValidForMY
1653            End Get
1654        End Property
1655
1656        '''
1657        ' ValidateForScenario
1658        ' Checks settings regarding this vehicle visi-vi the specified scenario.
1659        '
1660        Public Function ValidateForScenario(ByVal iScenario As Integer) As Boolean
1661            Me.ValidForScenario = Me.ValidOverAll
1662
1663            If Me.ValidOverAll Then
1664                If CAFE.Scenarios.Scenarios(iScenario).LtRegClassAttribute.ToUpper = "AREA" Then
1665                    ' must check wheelbase and track widths [KG:  changed 01/25/05 for footprint-based system]
1666                    '[KG:  changed following if/then 01/25/05 for footprint-based system]
1667                    If Me.ValidateDouble(Me.Wheelbase) AndAlso _
1668                       Me.ValidateDouble(Me.TrackWidthFront) AndAlso _
1669                       Me.ValidateDouble(Me.TrackWidthRear) Then
1670                        '
1671                        Me.ValidForScenario = True
1672                    Else
1673                        ' Ok, let's give it another chance. If it's a sedan, convertable, or wagon,
1674                        ' then we'll set the footprint to some average value and call it a day.
1675                        Dim VehStyle As String = Me.Style.Trim
1676                        Select Case VehStyle.ToUpper
1677                            Case "WAGON", "SEDAN", "CONVERTIBLE"
1678                                If Me.ValidateDouble(Me.Wheelbase) = False Then
1679                                    Me.Wheelbase = 107.9
```

```
1680                            End If
1681                            If Me.ValidateDouble(Me.TrackWidthFront) = False Then
1682                                Me.TrackWidthFront = 60.7
1683                            End If
1684                            If Me.ValidateDouble(Me.TrackWidthRear) = False Then
1685                                Me.TrackWidthRear = 60.7
1686                            End If
1687                            Me.ValidForScenario = True
1688                        Case Else
1689                            Me.ValidForScenario = False
1690                    End Select
1691                End If
1692            End If
1693        End If
1694
1695        Return Me.ValidForScenario
1696    End Function
1697
1698    '''
1699    ' ValidateForMY
1700    ' Checks settings regarding this vehicle visi-vi the specified model year.
1701    '
1702    Public Function ValidateForMY(ByVal iScenario As Integer, ByVal MY As Integer) As
        Boolean
1703        Me.ValidForMY = Me.ValidOverAll
1704
1705        If Me.ValidOverAll Then
1706            Dim iMY As Integer = CAFE.GMYI(MY)
1707
1708            If iMY >= Me.Sales.Length OrElse _
1709               iMY >= Me.MSRP.Length OrElse _
1710               iMY >= Me.Price.Length Then
1711                '
1712                Me.ValidForMY = False
1713            Else
1714                Me.ValidForMY = (Me.Sales(iMY) > 0 AndAlso _
1715                                 Me.MSRP(iMY) > 0 AndAlso _
1716                                 Me.Price(iMY) > 0)
1717            End If
1718        End If
1719
1720        Return Me.ValidForMY
1721    End Function
1722
1723 #End Region
1724
1725
1726 #Region "Market Simulation Support"
1727    Public Function GetFuelValueToBuyer(ByVal iMY As Integer) As Double
1728        ' Calculate value buyer places on future fuel outlays.
1729        If Not CAFE.OtherSettings.RunEffectsModel Then
1730            Return 0
1731        End If
1732
1733        ' Initialization
1734        Dim FuelValueToBuyer As Double = 0
1735        Dim i As Integer
1736        Dim FuelEconomy As Double = Me.FuelEconomy
1737        Dim Gap As Double = CAFE.Parameters.gapTestOnRoad
1738        Dim DiscountRate As Double = CAFE.Parameters.discountRate
1739        Dim Payback As Double = Math.Min(25, CAFE.Parameters.Payback)
1740        Dim FuelPrice() As Double
1741        Dim FuelType As String = UCase(CAFE.Engines(Me.TrueEngineCode).Fuel)
1742        If FuelType = "D" Then
1743            ' diesel
1744            FuelPrice = CAFE.Parameters.fuelPriceDiesel
1745        Else
1746            ' gas
1747            FuelPrice = CAFE.Parameters.fuelPriceGas
1748        End If
```

```
1749            Dim SurvRate(,) As Double
1750            SurvRate = GetSurvivalRate()
1751            Dim MilesDriven(,) As Double
1752            MilesDriven = GetMilesDriven()
1753
1754            ' Processing
1755            If Payback > 0 Then
1756                For i = 0 To Math.Floor(Payback)
1757                    FuelValueToBuyer += SurvRate(i, 0) * MilesDriven(i, 0) * FuelPrice(Math.
        Min(i + iMY, 25)) / _
1758                                        (FuelEconomy * (1 - Gap) * (1 + DiscountRate) ^ (i + 0
        .5))
1759                Next i
1760                ' add value of partial year (if any, such as 0.5 of 5th year for 4.5-year
        payback)
1761                i = Math.Floor(Payback)
1762                FuelValueToBuyer += (Payback - i) * SurvRate(i, 0) * MilesDriven(i, 0) * Math.
        Min(i + iMY, 25) / _
1763                                    (FuelEconomy * (1 - Gap) * (1 + DiscountRate) ^ (i + 0.5))
1764
1765            Else
1766                FuelValueToBuyer = 0
1767            End If
1768
1769            ' Termination
1770            Return FuelValueToBuyer
1771        End Function ' GetFuelValueToBuyer
1772    #End Region ' Market Simulation Support
1773
1774
1775    #Region "TechOverRide Support Routines"
1776        Public Function IsTechOverRideForce(ByVal TechIdx As Integer) As Boolean
1777            Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Force
1778        End Function
1779        Public Function IsTechOverRideForce(ByVal TechAbbreviation As String) As Boolean
1780            Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Force
1781        End Function
1782        Public Function IsTechOverRideSkip(ByVal TechIdx As Integer) As Boolean
1783            Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Skip
1784        End Function
1785        Public Function IsTechOverRideSkip(ByVal TechAbbreviation As String) As Boolean
1786            Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Skip
1787        End Function
1788        Public Function IsTechOverRideUsed(ByVal TechIdx As Integer) As Boolean
1789            Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.Used
1790        End Function
1791        Public Function IsTechOverRideUsed(ByVal TechAbbreviation As String) As Boolean
1792            Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.Used
1793        End Function
1794        Public Function IsTechOverRideNone(ByVal TechIdx As Integer) As Boolean
1795            Return GetTechOverRide(TechIdx) = CAFE.TechOverRide.None
1796        End Function
1797        Public Function IsTechOverRideNone(ByVal TechAbbreviation As String) As Boolean
1798            Return GetTechOverRide(TechAbbreviation) = CAFE.TechOverRide.None
1799        End Function
1800        '''
1801        ' Get the TechOverRide setting.
1802        ' This is rather brute-force with a complete lack of elegance,
1803        ' maybe we'll clean this up someday -- but don't count on it
1804        ' jvs 15 oct 04
1805        Public Function GetTechOverRide(ByVal TechIdx As Integer) As CAFE.TechOverRide
1806            Dim Abbr As String
1807            ' Get the abbreviation using NRC Vehicle Class 0 (the abbrv are the same for all)
1808            Abbr = CAFE.Technologies.VehicleClasses(0).Technology(TechIdx).Abbr
1809            Return GetTechOverRide(Abbr)
1810        End Function
1811        Public Function GetTechOverRide(ByVal TechAbbreviation As String) As CAFE.TechOverRide
1812            Dim RtnCode As CAFE.TechOverRide = CAFE.TechOverRide.None
1813            Select Case TechAbbreviation
1814                Case "ROLL"
```

```
1815                RtnCode = Me.TechOverROLL
1816            Case "LDB"
1817                RtnCode = Me.TechOverLDB
1818            Case "FAD"
1819                RtnCode = Me.TechOverFAD
1820            Case "EPS"
1821                RtnCode = Me.TechOverEPS
1822            Case "EAI"
1823                RtnCode = Me.TechOverEAI
1824            Case "AER"
1825                RtnCode = Me.TechOverAER
1826            Case "42V"
1827                RtnCode = Me.TechOver42V
1828            Case "ISG"
1829                RtnCode = Me.TechOverISG
1830            Case "MS1"
1831                RtnCode = Me.TechOverMS1
1832            Case "MS2"
1833                RtnCode = Me.TechOverMS2
1834            Case "MS3"
1835                RtnCode = Me.TechOverMS3
1836            Case "MSX"
1837                RtnCode = Me.TechOverMSX
1838        End Select
1839        Return RtnCode
1840    End Function
1841 #End Region
1842
1843
1844 End Class
1845
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This class stores data that will be outputed in reports
'''
Public Class VehicleStackClass
    Inherits CafeStackClass

    ' vehicle's data at the specific model year
    Public ModelIndex As Integer                        ' 0
    Public ModelID As Integer                           ' 1
    '
    Public Manufacturer As String                       ' 2
    Public Model As String                              ' 3
    Public NamePlate As String                          ' 4
    '
    Public RegulatoryClass As Integer                   ' 5
    Public TotalSalesInitial As Double                  ' 6
    Public TotalSalesFinal As Double                    ' 7
    Public MSRPInitial As Double                        ' 8
    Public PriceInitial As Double                       ' 9
    Public FuelEconomyInitial As Double                 ' 10
    Public FuelEconomyFinal As Double                   ' 11
    Public CurbWeightInitial As Double                  ' 12
    Public CurbWeightFinal As Double                    ' 13
    Public Area As Double                               ' 14
    Public EngineID As Integer                          ' 15
    Public EngineFuelType As String                     ' 16
    Public EngineDisplacement As Double                 ' 17
    Public EngineCylinderCount As Integer               ' 18
    Public TransmissionID As Integer                    ' 19
    Public TransmissionType As String                       ' 20
    Public IncurredTechnologyCostsPerVehicle As Double      ' 21
    Public PriceIncrease As Double                          ' 22
    Public TotalIncurredTechnologyCostsRPE As Double        ' 23
    Public TotalPriceIncrease As Double                     ' 24
    Public Utilization As TechnologiesStackClass        ' 25
    Public Applicability() As Boolean                   ' 26
    Protected TotalUSEmploymentHours As Double          ' employment hours is protected because
    '                                                   ' we don't want it to be part of the stack
    Public TotalUSEmploymentJobs As Double              ' 27

    Public Sub New()
        MyBase.New()
    End Sub

    Public Sub New(ByVal MY As Integer, ByVal veh As VehiclesClass)
        Me.New()
        CreateStack(MY, veh)
    End Sub

    Public Function CreateStack(ByVal MY As Integer, ByVal veh As VehiclesClass) As VehicleStackClass
        Dim i As Integer
        Dim eng As EnginesClass
        Dim trn As TransmissionsClass
        Dim MYIdx As Integer
        Dim trnType As String

        eng = CAFE.Engines(veh.TrueEngineCode)                          ' get vehicle's engine
        trn = CAFE.Transmissions(veh.TrueTransmissionCode)              ' get vehicle's transmission
        MYIdx = CAFE.GMYI(MY)                                           ' get model year index

        Me.MY = MY
        Me.Valid = veh.IsValid()
        Me.ModelIndex = veh.MyIndexNumber
        Me.ModelID = veh.Number
        Me.Manufacturer = veh.Manufacturer
        Me.Model = veh.Model
        Me.NamePlate = veh.NamePlate
```

```
68          Me.RegulatoryClass = veh.RegClass
69          Me.TotalSalesInitial = veh.InitialSales(MYIdx)
70          Me.TotalSalesFinal = veh.Sales(MYIdx)
71          Me.MSRPInitial = veh.InitialMSRP(MYIdx)
72          Me.PriceInitial = veh.InitialPrice(MYIdx)
73          Me.FuelEconomyInitial = veh.InitialFuelEconomy
74          Me.FuelEconomyFinal = veh.FuelEconomy
75          Me.CurbWeightInitial = veh.InitialCurbWeight
76          Me.CurbWeightFinal = veh.CurbWeight
77          Me.Area = veh.OverAllArea
78          Me.EngineID = veh.EngineCode
79          Me.EngineFuelType = eng.Fuel
80          Me.EngineDisplacement = eng.Displacement
81          Me.EngineCylinderCount = eng.Cylinders
82          Me.TransmissionID = veh.TransmissionCode
83          trnType = UCase(trn.Type)
84          If trnType.StartsWith("CVT") Then
85              Me.TransmissionType = "CVT"
86          ElseIf trnType = "C" AndAlso UCase(trn.Control) = "A" Then
87              Me.TransmissionType = "AST"
88          Else
89              Me.TransmissionType = trn.Control & trn.NumberForwardGears
90          End If
91          Me.IncurredTechnologyCostsPerVehicle = veh.TechCost
92          Me.PriceIncrease = veh.RegCost
93          Me.TotalIncurredTechnologyCostsRPE = veh.TechCost * veh.Sales(MYIdx) / 1000      '  ↙
        (in thousands)
94          Me.TotalPriceIncrease = veh.RegCost * veh.Sales(MYIdx) / 1000                    '  ↙
        increase in purchase cost (in thousands)
95          TechnologiesStackClass.Copy(Me.Utilization, veh.TechUsedStack)
96          If Not IsNothing(veh.TechEnabled) Then
97              ReDim Me.Applicability(veh.TechEnabled.Length - 1)
98              For i = 0 To veh.TechEnabled.Length - 1
99                  Me.Applicability(i) = veh.TechEnabled(i)
100             Next i
101         End If
102
103
104
105
106         Return Me
107     End Function
108
109     Public Sub Copy(ByRef dst As VehicleStackClass)
110         Me.Copy(dst, Me)
111     End Sub
112
113     Public Shared Sub Copy(ByRef dst As VehicleStackClass, ByRef src As VehicleStackClass)
114         Dim i As Integer
115
116         dst = New VehicleStackClass
117
118         dst.Scenario = src.Scenario
119         dst.MY = src.MY
120         dst.Valid = src.Valid
121         dst.ModelIndex = src.ModelIndex
122         dst.ModelID = src.ModelID
123         dst.Manufacturer = src.Manufacturer
124         dst.Model = src.Model
125         dst.NamePlate = src.NamePlate
126         dst.RegulatoryClass = src.RegulatoryClass
127         dst.TotalSalesInitial = src.TotalSalesInitial
128         dst.TotalSalesFinal = src.TotalSalesFinal
129         dst.MSRPInitial = src.MSRPInitial
130         dst.PriceInitial = src.PriceInitial
131         dst.FuelEconomyInitial = src.FuelEconomyInitial
132         dst.FuelEconomyFinal = src.FuelEconomyFinal
```

```
        dst.CurbWeightInitial = src.CurbWeightInitial
        dst.CurbWeightFinal = src.CurbWeightFinal
        dst.Area = src.Area
        dst.EngineID = src.EngineID
        dst.EngineFuelType = src.EngineFuelType
        dst.EngineDisplacement = src.EngineDisplacement
        dst.EngineCylinderCount = src.EngineCylinderCount
        dst.TransmissionID = src.TransmissionID
        dst.TransmissionType = src.TransmissionType
        dst.IncurredTechnologyCostsPerVehicle = src.IncurredTechnologyCostsPerVehicle
        dst.PriceIncrease = src.PriceIncrease
        dst.TotalIncurredTechnologyCostsRPE = src.TotalIncurredTechnologyCostsRPE
        dst.TotalPriceIncrease = src.TotalPriceIncrease
        TechnologiesStackClass.Copy(dst.Utilization, src.Utilization)
        If Not IsNothing(src.Applicability) Then
            ReDim dst.Applicability(src.Applicability.Length - 1)
            For i = 0 To src.Applicability.Length - 1
                dst.Applicability(i) = src.Applicability(i)
            Next i
        End If
        dst.TotalUSEmploymentHours = src.TotalUSEmploymentHours
        dst.TotalUSEmploymentJobs = src.TotalUSEmploymentJobs
    End Sub

    Public Overrides Function ToArray() As Object()
        Dim size As Integer = 28
        Dim stack(size - 1) As Object

        stack(0) = Me.ModelIndex
        stack(1) = Me.ModelID
        stack(2) = Me.Manufacturer
        stack(3) = Me.Model
        stack(4) = Me.NamePlate
        stack(5) = Me.RegulatoryClass
        stack(6) = Me.TotalSalesInitial
        stack(7) = Me.TotalSalesFinal
        stack(8) = Me.MSRPInitial
        stack(9) = Me.PriceInitial
        stack(10) = Me.FuelEconomyInitial
        stack(11) = Me.FuelEconomyFinal
        stack(12) = Me.CurbWeightInitial
        stack(13) = Me.CurbWeightFinal
        stack(14) = Me.Area
        stack(15) = Me.EngineID
        stack(16) = Me.EngineFuelType
        stack(17) = Me.EngineDisplacement
        stack(18) = Me.EngineCylinderCount
        stack(19) = Me.TransmissionID
        stack(20) = Me.TransmissionType
        stack(21) = Me.IncurredTechnologyCostsPerVehicle
        stack(22) = Me.PriceIncrease
        stack(23) = Me.TotalIncurredTechnologyCostsRPE
        stack(24) = Me.TotalPriceIncrease
        stack(25) = Me.Utilization
        stack(26) = Me.Applicability
        'stack(21) = Me.TotalUSEmploymentHours                    ' employment hours should not be part of the stack
        stack(27) = Me.TotalUSEmploymentJobs

        Return stack
    End Function

End Class

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This class stores data per model year
'''
Public Class VehiclesStackClass
    Inherits CafeStacksClass

```

```
202     Public Overrides Function Push(ByVal vsc As CafeStackClass) As Boolean
203         ' create and copy the new VehicleStackClass
204         Dim entry As VehicleStackClass
205         VehicleStackClass.copy(entry, vsc)
206         '
207         ' store it to the stack
208         Me.MyStack(Me.MySize) = entry
209         Me.MySize += 1
210
211         If Me.MySize >= Me.MyStack.Length Then
212             ' array at max capacity; expand the array
213             ReDim Preserve Me.MyStack(Me.MySize + (Me.CapacityIncrement - 1))
214         End If
215
216         Return True
217     End Function
218
219 End Class
220
```

```
Public Class VehicleTechnologiesClass
    ' All the technologies for all vehicle classes
    Public nVehicleClasses As Integer
    Public VehicleClasses() As TechnologiesClass

    Public Sub New()
        nVehicleClasses = 0
    End Sub
    Public Sub Add()
        ReDim Preserve VehicleClasses(nVehicleClasses)
        VehicleClasses(nVehicleClasses) = New TechnologiesClass
        nVehicleClasses += 1
    End Sub
    Public Sub Add(ByRef newTechnologies As TechnologiesClass)
        ReDim Preserve VehicleClasses(nVehicleClasses)
        VehicleClasses(nVehicleClasses) = New TechnologiesClass
        newTechnologies.Copy(VehicleClasses(nVehicleClasses))
        nVehicleClasses += 1
    End Sub
    Public Sub Copy(ByRef Dst As VehicleTechnologiesClass)
        Copy(Dst, Me)
    End Sub
    Public Shared Sub Copy(ByRef Dst As VehicleTechnologiesClass, ByRef Src As
    VehicleTechnologiesClass)
        Dim i As Integer
        Dst.nVehicleClasses = Src.nVehicleClasses
        ReDim Dst.VehicleClasses(Src.nVehicleClasses - 1)
        For i = 0 To Src.nVehicleClasses - 1
            Dst.VehicleClasses(i) = New TechnologiesClass
            Src.VehicleClasses(i).Copy(Dst.VehicleClasses(i))
        Next
    End Sub
    Public Function GetNumClasses() As Integer
        Return nVehicleClasses
    End Function
    Public Function GetNumTechnologies(ByVal VehicleClass As Integer) As Integer
        Return GetNumTechnologies()
    End Function
    Public Function GetNumTechnologies() As Integer
        Return TechnologiesClass.nTotalTechnologies
    End Function
    Public Function GetNumTransmissionTechnologies(ByVal VehicleClass As Integer)
        Return GetNumTransmissionTechnologies()
    End Function
    Public Function GetNumTransmissionTechnologies()
        Return TechnologiesClass.nTranTechnologies
    End Function
    Public Function GetNumEngineTechnologies(ByVal VehicleClass As Integer)
        Return GetNumEngineTechnologies()
    End Function
    Public Function GetNumEngineTechnologies()
        Return TechnologiesClass.nEngTechnologies
    End Function
    Public Function GetNumAccLoadTechnologies(ByVal VehicleClass As Integer)
        Return GetNumAccLoadTechnologies()
    End Function
    Public Function GetNumAccLoadTechnologies()
        Return TechnologiesClass.nAccLoadTechnologies
    End Function
    Public Function GetNumDLRTechnologies(ByVal VehicleClass As Integer)
        Return GetNumDLRTechnologies()
    End Function
    Public Function GetNumDLRTechnologies()
        Return TechnologiesClass.nDLRTechnologies
    End Function
    Public Function GetNumOtherTechnologies(ByVal VehicleClass As Integer)
        Return GetNumOtherTechnologies()
    End Function
    Public Function GetNumOtherTechnologies()
        Return TechnologiesClass.nOtherTechnologies
```

```
70      End Function
71      Public Function GetCost(ByVal TechClass As Integer, ByVal TechIdx As Integer) As Double
72          Return VehicleClasses(TechClass).GetCost(TechIdx)
73      End Function
74      Public Function GetImprv(ByVal TechClass As Integer, ByVal TechIdx As Integer) As
        Double
75          Return VehicleClasses(TechClass).GetImprv(TechIdx)
76      End Function
77      Public Function IsMaterialSubstitution(ByVal TechClass As Integer, ByVal TechIdx As
        Integer) As Boolean
78          ' This flag is only set for tech class 0, so we'll look there...
79          Return VehicleClasses(0).Technology(TechIdx)._IsMaterialSubstitution
80      End Function
81      Public Function IsDieselization(ByVal TechClass As Integer, ByVal TechIdx As Integer)
        As Boolean
82          ' This flag is only set for tech class 0, so we'll look there...
83          Return VehicleClasses(0).Technology(TechIdx)._IsDieselization
84      End Function
85      Public Function IsRemovable(ByVal TechClass As Integer, ByVal TechIdx As Integer) As
        Boolean
86          ' A removable technology is one which can be removed after it has been installed
        (only in the same MY)
87          ' This flag is only set for tech class 0, so we'll look there...
88          Return VehicleClasses(0).Technology(TechIdx).Removable
89      End Function
90      Public Function GetKWeight(ByVal TechClass As Integer, ByVal TechIdx As Integer) As
        Double
91          If IsMaterialSubstitution(TechClass, TechIdx) Then
92              Return Me.VehicleClasses(TechClass).Technology(TechIdx).kWeight
93          Else
94              Return Double.NaN
95          End If
96      End Function
97
98      ' Sets the NRC Class for the vehicle.
99      Public Sub SetNRC_Class(ByRef Veh As VehiclesClass, ByVal Eng As EnginesClass)
100         ' gets the NRC_Class of a vehicle at the specified 'VehIdx' index
101         Dim NRC_Class As String
102         Dim VehStyle As String = (UCase(Veh.Style)).Trim()
103         Dim VehClass As String = (UCase(Veh.VehClass)).Trim()
104         Dim VehCylinders As Integer = Eng.Cylinders
105
106         Select Case VehStyle
107             Case "UTILITY"
108                 If VehCylinders <= 4 Then
109                     NRC_Class = "SUV-Small"
110                 ElseIf VehCylinders <= 6 Then
111                     NRC_Class = "SUV - Mid"
112                 Else
113                     NRC_Class = "SUV - Large"
114                 End If
115             Case "VAN"
116                 NRC_Class = "MiniVan"
117             Case "PICKUP"
118                 If VehCylinders <= 6 Then
119                     NRC_Class = "Pickup-Small"
120                 Else
121                     NRC_Class = "Pickup - Large"
122                 End If
123             Case "WAGON", "SEDAN", "CONVERTIBLE"
124                 Select Case VehClass
125                     Case "MINI-COMPACT", "SUBCOMPACT", "TWO SEATER"
126                         NRC_Class = "Subcompact"
127                     Case "COMPACT"
128                         NRC_Class = "Compact"
129                     Case "MIDSIZE"
130                         NRC_Class = "Midsize"
131                     Case "LARGE", "PASSENGER VAN"
132                         NRC_Class = "Large"
133                     Case Else
```

```
134                         ' all other vehicle styles
135                         NRC_Class = "OTHER"
136                 End Select
137             Case Else
138                 NRC_Class = "OTHER"
139         End Select
140         Veh.NRC_Class = NRC_Class
141
142         ' Get technology class index based on class name
143         ' returns -1 if not found.
144         Dim nrcidx As Integer = -1
145         Dim i As Integer
146         For i = 0 To Me.nVehicleClasses - 1
147             If UCase((Me.VehicleClasses(i).NRC_ClassName).Trim()) = UCase((NRC_Class).Trim ↙
    ()) Then
148                 nrcidx = i
149                 Exit For
150             End If
151         Next
152         Veh.NRC_ClassIdx = nrcidx
153     End Sub
154
155     Public Sub SetTechnologyEnable(ByVal MY As Integer, ByVal vehidx As Integer)
156         ' Set technology Enable
157         ' Sets the technology enable array for a vehicle' Entry args:
158         '   MY = Model year
159         '   VehIdx = Vehicle Index
160
161         ReDim CAFE.Vehicles(vehidx).TechEnabled(TechnologiesClass.nTotalTechnologies - 1)
162
163         For t As Integer = 0 To TechnologiesClass.nTotalTechnologies - 1
164             Dim nrc As Integer = CAFE.Vehicles(vehidx).NRC_ClassIdx
165             If nrc <> -1 AndAlso VehicleClasses(nrc).Technology(t).YearAvailable <= MY Then
166                 ' Get tech overrides from the veh/eng/trans database
167                 Dim TOR As CAFE.TechOverRide
168                 If VehicleClasses(nrc).Technology(t).TechType.ToUpper = "ENGMOD" Then
169                     Dim ei As Integer = CAFE.Vehicles(vehidx).TrueEngineCode
170                     TOR = CAFE.Engines(ei).GetTechOverRide(t)
171                 ElseIf VehicleClasses(nrc).Technology(t).TechType.ToUpper = "TRMOD" Then
172                     Dim ti As Integer = CAFE.Vehicles(vehidx).TrueTransmissionCode
173                     TOR = CAFE.Transmissions(ti).GetTechOverRide(t)
174                 Else
175                     TOR = CAFE.Vehicles(vehidx).GetTechOverRide(t)
176                 End If
177
178                 ' Check the path settings.
179                 Select Case CAFE.PathSettings
180                     Case CAFE.PathsEnum.Path1
181                         CAFE.Vehicles(vehidx).TechEnabled(t) = VehicleClasses(nrc). ↙
    Technology(t).Path1
182                     Case CAFE.PathsEnum.Path2
183                         CAFE.Vehicles(vehidx).TechEnabled(t) = VehicleClasses(nrc). ↙
    Technology(t).Path2
184                     Case CAFE.PathsEnum.Path3
185                         CAFE.Vehicles(vehidx).TechEnabled(t) = VehicleClasses(nrc). ↙
    Technology(t).Path3
186                 End Select
187                 ' Filter it further using TechApplies
188                 CAFE.Vehicles(vehidx).TechEnabled(t) = CAFE.Vehicles(vehidx).TechEnabled(t) ↙
     And CAFE.Vehicles(vehidx).TechApplies(t)
189
190                 ' Special Cased -- MS2 needs to have MS1 in place, MS3 needs to have MS2 in ↙
     place
191                 If t = TechnologiesClass.TechIndexes.tMS_MS2 Then
192                     CAFE.Vehicles(vehidx).TechMS2Enablable = CAFE.Vehicles(vehidx). ↙
    TechEnabled(t)
193                     CAFE.Vehicles(vehidx).TechEnabled(t) = _
194                         CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes. ↙
    tMS_MS1) And CAFE.Vehicles(vehidx).TechMS2Enablable
195                 End If
```

```
196                 If t = TechnologiesClass.TechIndexes.tMS_MS3 Then
197                     CAFE.Vehicles(vehidx).TechMS3Enablable = CAFE.Vehicles(vehidx).
    TechEnabled(t)
198                     CAFE.Vehicles(vehidx).TechEnabled(t) =
199                         CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.
    tMS_MS2) And CAFE.Vehicles(vehidx).TechMS3Enablable
200                 End If
201
202                 ' Add in the over-ride ingredients
203                 If CAFE.Vehicles(vehidx).TechEnabled(t) Then
204                     If TOR = CAFE.TechOverRide.Skip Or TOR = CAFE.TechOverRide.Used Then
205                         CAFE.Vehicles(vehidx).TechEnabled(t) = False
206                     End If
207                 Else
208                     If TOR = CAFE.TechOverRide.Force Then
209                         CAFE.Vehicles(vehidx).TechEnabled(t) = True
210                     End If
211                 End If
212             Else
213                 CAFE.Vehicles(vehidx).TechEnabled(t) = False
214             End If
215         Next t
216
217         ' Sort through the tech overrides from the veh database
218         For t As Integer = 0 To TechnologiesClass.nTotalTechnologies - 1
219
220         Next t
221         ' engines
222     End Sub
223     Public Sub InitTechnologyApplies(ByVal vehidx As Integer)
224         Dim engidx As Integer = CAFE.Vehicles(vehidx).TrueEngineCode
225         Dim tranidx As Integer = CAFE.Vehicles(vehidx).TrueTransmissionCode
226
227         ReDim CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.nTotalTechnologies - 1)
228         ReDim CAFE.Vehicles(vehidx).TechReserved(TechnologiesClass.nTotalTechnologies - 1)
229         ReDim CAFE.Vehicles(vehidx).TechAppliedByModel(TechnologiesClass.nTotalTechnologies
     - 1)
230         ReDim CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.nTotalTechnologies - 1)
231
232         ' Initially, set applicability per chosen NRC Path settings.
233         ' NOTE:  The TechPathOverride sub allows the user to force technologies to be
    enabled
234         ' or disabled on an engine-, transmission-, or vehicle-specific basis,
    notwithstanding
235         ' (i.e., overriding) the selected technology path.
236         For t As Integer = 0 To TechnologiesClass.nTotalTechnologies - 1
237             Dim nrc As Integer = CAFE.Vehicles(vehidx).NRC_ClassIdx
238             If nrc <> -1 Then
239                 ' Get tech overrides from the veh/eng/trans database
240                 Dim TOR As CAFE.TechOverRide
241                 If VehicleClasses(nrc).Technology(t).TechType.ToUpper = "ENGMOD" Then
242                     Dim ei As Integer = CAFE.Vehicles(vehidx).TrueEngineCode
243                     TOR = CAFE.Engines(ei).GetTechOverRide(t)
244                 ElseIf VehicleClasses(nrc).Technology(t).TechType.ToUpper = "TRMOD" Then
245                     Dim ti As Integer = CAFE.Vehicles(vehidx).TrueTransmissionCode
246                     TOR = CAFE.Transmissions(ti).GetTechOverRide(t)
247                 Else
248                     TOR = CAFE.Vehicles(vehidx).GetTechOverRide(t)
249                 End If
250
251                 Select Case CAFE.PathSettings
252                     Case CAFE.PathsEnum.Path1
253                         CAFE.Vehicles(vehidx).TechApplies(t) = VehicleClasses(nrc).
    Technology(t).Path1
254                     Case CAFE.PathsEnum.Path2
255                         CAFE.Vehicles(vehidx).TechApplies(t) = VehicleClasses(nrc).
    Technology(t).Path2
256                     Case CAFE.PathsEnum.Path3
257                         CAFE.Vehicles(vehidx).TechApplies(t) = VehicleClasses(nrc).
    Technology(t).Path3
```

```
258                 End Select
259                 ' Add in the over-ride ingredients
260                 If CAFE.Vehicles(vehidx).TechApplies(t) Then
261                     If TOR = CAFE.TechOverRide.Skip Or TOR = CAFE.TechOverRide.Used Then
262                         CAFE.Vehicles(vehidx).TechApplies(t) = False
263                     End If
264                 Else
265                     If TOR = CAFE.TechOverRide.Force Then
266                         CAFE.Vehicles(vehidx).TechApplies(t) = True
267                     End If
268                 End If
269             Else
270                 CAFE.Vehicles(vehidx).TechApplies(t) = False
271             End If
272         Next
273
274         ' Use other inputs to identify used and otherwise unavailable technologies.
275
276         ' Engine Technologies
277
278         ' Low friction lubricants (LUB)
279         ' If engine oil less viscous than 10W30, assume low friction lubricants used.
280         If CAFE.Engines(engidx).TechOverLUB = CAFE.TechOverRide.Used OrElse _
281         (CAFE.Engines(engidx).EngineOilViscosity.ToUpper <> "" AndAlso _
282         CAFE.Engines(engidx).EngineOilViscosity.ToUpper <> "10W40" AndAlso _
283         CAFE.Engines(engidx).EngineOilViscosity.ToUpper <> "10W30" AndAlso _
284         CAFE.Engines(engidx).EngineOilViscosity.ToUpper <> "15W40" AndAlso _
285         CAFE.Engines(engidx).EngineOilViscosity.ToUpper <> "LUB1" AndAlso _
286         CAFE.Engines(engidx).EngineOilViscosity.ToUpper <> "LUB2") Then
287             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngLUB) = True
288             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngLUB) =
    False
289         End If
290
291         ' Engine friction reduction (EFR)
292         If CAFE.Engines(engidx).TechOverEFR = CAFE.TechOverRide.Used Then
293             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngEFR) = True
294             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngEFR) =
    False
295         End If
296
297         ' Multivalve OHC
298         ' If more than 2 valves per cylinder, assume multivalve OHC used.
299         If CAFE.Engines(engidx).TechOverOHC = CAFE.TechOverRide.Used OrElse _
300         CAFE.Engines(engidx).ValvesCylinder > 2 Then
301             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngOHC) = True
302             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngOHC) =
    False
303         End If
304
305         ' Variable valve timing (VVT)
306         ' If valve timing not fixed, assume VVT used.
307         If CAFE.Engines(engidx).TechOverVVT = CAFE.TechOverRide.Used OrElse _
308         (CAFE.Engines(engidx).ValveActuationTiming.ToUpper <> "" AndAlso _
309         CAFE.Engines(engidx).ValveActuationTiming.ToUpper <> "F") Then
310             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngVVT) = True
311             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVVT) =
    False
312         End If
313         ' Do not apply to engines with displacement greater than 4.7 liters.
314         If CAFE.Engines(engidx).Displacement > 4.7 Then
315             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVVT) =
    False
316         End If
317
318         ' Variable valve lift and timing (VVLT)
319         ' If valve lift not fixed, assume VVLT used.
320         If CAFE.Engines(engidx).TechOverVVLT = CAFE.TechOverRide.Used OrElse _
321         (CAFE.Engines(engidx).ValveLift.ToUpper <> "" AndAlso _
322         CAFE.Engines(engidx).ValveLift.ToUpper <> "F") Then
```

```
323             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngVVLT) = True
324             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVVT) =
    False
325             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVVLT) =
    False
326         End If
327         ' Do not apply to engines with displacement of more than 3.0 liters.
328         If CAFE.Engines(engidx).Displacement > 3 Then
329             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVVLT) =
    False
330         End If
331
332         ' Intake valve throttling (IVT)
333         ' If valve lift continuously variable but valvetrain design not camless, assume IVT
     used.
334         If CAFE.Engines(engidx).TechOverIVT = CAFE.TechOverRide.Used OrElse _
335         (CAFE.Engines(engidx).ValveLift.ToUpper = "CV" AndAlso _
336         CAFE.Engines(engidx).ValvetrainDesign.ToUpper <> "" AndAlso _
337         CAFE.Engines(engidx).ValvetrainDesign.ToUpper <> "C" AndAlso _
338         CAFE.Engines(engidx).TechOverCVA <> CAFE.TechOverRide.Used) Then
339             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngIVT) = True
340             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngIVT) =
    False
341         End If
342
343         ' Camless valve actuation (CVA)
344         ' If valvetrain design camless, assume camless valve actuation used.
345         If CAFE.Engines(engidx).TechOverCVA = CAFE.TechOverRide.Used OrElse _
346         CAFE.Engines(engidx).ValvetrainDesign.ToUpper = "C" Then
347             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngCVA) = True
348             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngCVA) =
    False
349         End If
350
351         ' Cylinder deactivation (DSP)
352         ' If amount of deactivation > 0, assume cylinder deactivation used.
353         ' If base engine has cylinder deactivation, skip multivalve OHC, VVT, and VVLT.
354         If CAFE.Engines(engidx).TechOverDISP = CAFE.TechOverRide.Used OrElse _
355         CAFE.Engines(engidx).Deactivation > 0 Then
356             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngDSP) = True
357             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngDSP) =
    False
358             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngOHC) =
    False
359             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVVLT) =
    False
360             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVVT) =
    False
361         End If
362         ' Do not apply cylinder deactivation to engines with fewer than 6 cylinders.
363         ' Do not apply to engines with VVT or VVLT.
364         If CAFE.Engines(engidx).Cylinders < 6 OrElse _
365         (CAFE.Engines(engidx).ValveActuationTiming <> "" AndAlso _
366         CAFE.Engines(engidx).ValveActuationTiming <> "F") OrElse _
367         (CAFE.Engines(engidx).ValveLift <> "" AndAlso _
368         CAFE.Engines(engidx).ValveLift <> "F") Then
369             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngDSP) =
    False
370         End If
371
372         ' Spark-ignited direct injecion (SIDI)
373         ' If cycle is "O" (Otto) and fuel system is "DI", then assume SIDI.
374         If CAFE.Engines(engidx).TechOverSIDI = CAFE.TechOverRide.Used OrElse _
375         (CAFE.Engines(engidx).Cycle.ToUpper = "O" And _
376         CAFE.Engines(engidx).FuelSystem.ToUpper = "DI") Then
377             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngSIDI) = True
378             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngSIDI) =
    False
379         End If
380         ' Do not apply to engines with more than 6 cylinders.
```

```
381         If CAFE.Engines(engidx).Cylinders > 6 Then
382             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngSIDI) =
    False
383         End If
384
385         ' Engine supercharging and downsizing (SUP)
386         ' If aspiration turbocharged or supercharged, assume supercharging and downsizing
    used.
387         If CAFE.Engines(engidx).TechOverSUP = CAFE.TechOverRide.Used OrElse _
388         CAFE.Engines(engidx).Aspiration.ToUpper = "T" OrElse _
389         CAFE.Engines(engidx).Aspiration.ToUpper = "S" Then
390             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngSUP) = True
391             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngSUP) =
    False
392         End If
393
394         ' Variable compression ratio (VCR)
395         ' If min and max compression differ, assume variable compression ratio used.
396         If CAFE.Engines(engidx).TechOverVCR = CAFE.TechOverRide.Used OrElse _
397         (CAFE.Engines(engidx).MinCompressionRatio > 0 AndAlso _
398         CAFE.Engines(engidx).MaxCompressionRatio > 0 AndAlso _
399         CAFE.Engines(engidx).MinCompressionRatio <> CAFE.Engines(engidx).
    MaxCompressionRatio) Then
400             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngVCR) = True
401             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVCR) =
    False
402         End If
403
404         ' If Diesel cycle, turn off Otto cycle engine technologies.
405         ' TODO:  check applicability of supercharging, cylinder deactivation and
    dieselization
406         If CAFE.Engines(engidx).Fuel.ToUpper = "D" Then
407             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngOHC) =
    False
408             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVVT) =
    False
409             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVVLT) =
    False
410             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngSIDI) =
    False
411             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngSUP) =
    False
412             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngDSP) =
    False
413             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngIVT) =
    False
414             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngCVA) =
    False
415             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngVCR) =
    False
416             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngDSL) =
    False
417         End If
418
419         ' TODO: Decide whether to add routine that goes through all vehicles to force
420         '       consistency among vehicles that use the same engines.  For example, if VVT
421         '       appears to have been used by one vehicle among three that use a given
    engine,
422         '       should it be assumed that VVT is also present on the other two vehicles?
423         '       The answer to this question may be different with respect to applicability,
424         '       particularly if the model is allowed to "split" engine lines.  For example,
425         '       if VVT is not present on an engine and can only be applied to one of three
426         '       vehicles that use that engine, it may be appropriate not to declare VVT
427         '       inapplicable to that vehicle.
428
429         ' Transmission Technologies
430
431         ' Aggressive shift logic (ASL)
432         If CAFE.Transmissions(tranidx).TechOverASL = CAFE.TechOverRide.Used OrElse _
433         CAFE.Transmissions(tranidx).Logic = "A" Then
```

```
434             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tTrnASL) = True
435             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnASL) =
    False
436         End If
437
438         ' Automatic (5SP, 6SP) and continuously variable transmissions (CVT, ACV)
439         If CAFE.Transmissions(tranidx).Type.ToUpper = "T" Then
440             ' This transmission has a torque converter (i.e., it's an automatic).
441             If CAFE.Transmissions(tranidx).NumberForwardGears = 5 Then
442                 ' A5
443                 CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tTrn5SP) =
    True
444                 CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrn5SP) =
    False
445             ElseIf CAFE.Transmissions(tranidx).NumberForwardGears >= 6 Then
446                 ' A6:  Turn off A5, A6, CVT, and ACVT
447                 ' To do:  see if ASL should be skipped for A6 transmissions
448                 CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tTrn6SP) =
    True
449                 CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrn5SP) =
    False
450                 CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrn6SP) =
    False
451                 CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnCVT) =
    False
452                 CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnACV) =
    False
453                 ' CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnASL)
    = False
454             End If
455         End If
456         If Left(CAFE.Transmissions(tranidx).Type.ToUpper, 3) = "CVT" Then
457             ' CVT:  turn off other transmission technologies (except HEV)
458             ' TODO:  Decide whether to identify any pickup and midrange SUV CVTs as
    "advanced".
459             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tTrnCVT) = True
460             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrn5SP) =
    False
461             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrn6SP) =
    False
462             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnASL) =
    False
463             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnCVT) =
    False
464             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnACV) =
    False
465             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnAST) =
    False
466         End If
467         ' Do not apply CVT to frame pickups and SUVs with 4WD.
468         If (CAFE.Vehicles(vehidx).Style.ToUpper = "UTILITY" Or _
469         CAFE.Vehicles(vehidx).Style.ToUpper = "PICKUP") And _
470         (CAFE.Vehicles(vehidx).VehStructure.ToUpper = "FRAME" Or _
471         CAFE.Vehicles(vehidx).VehStructure.ToUpper = "LADDER") And _
472         (CAFE.Vehicles(vehidx).Drive.ToUpper = "4" Or _
473         CAFE.Vehicles(vehidx).Drive.ToUpper = "A") Then
474             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnCVT) =
    False
475             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnACV) =
    False
476         End If
477
478         ' Clutch (i.e., manual) transmission (AST)
479         If CAFE.Transmissions(tranidx).Type.ToUpper = "C" Then
480             ' This transmission has a clutch.
481             ' Assume only change would be to ASCT (a.k.a. ASMT, ACT, or AST)
482             ' NOTE:  For now (05/11/04), this refrains from using the previous approach,
    which
483             ' entailed numerous scaling factors discounting the effects of going to 5 or 6
484             ' speeds in a transmission with a torque converter.  If this simpler approach
```

```
485             ' is too limited, we may need to reconsider.
486             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrn5SP) =
    False
487             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrn6SP) =
    False
488             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnASL) =
    False
489             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnCVT) =
    False
490             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnACV) =
    False
491             If CAFE.Transmissions(tranidx).Control.ToUpper = "A" Then
492                 ' This is already an automatically shifted clutch transmission.
493                 CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tTrnAST) =
    True
494                 CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnAST) =
    False
495             End If
496         End If
497
498         ' TODO: Decide whether to add routine that goes through all vehicles to force
499         '       consistency among vehicles that use the same transmissions.
500         '       See discussion of this issue as it applies to engine technologies.
501
502         ' Other technologies
503
504         ' Improved rolling resistance (ROLL)
505         ' TODO:  Decide on maximum eligible Crr.
506         ' If CAFE.Vehicles(vehidx).TireRollingResistance < Crr_max then...
507         If CAFE.Vehicles(vehidx).TechOverROLL = CAFE.TechOverRide.Used Then
508             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tVehROLL) = True
509             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehROLL) =
    False
510         ElseIf (CAFE.Vehicles(vehidx).Style.ToUpper = "PICKUP" Or _
511             CAFE.Vehicles(vehidx).Style.ToUpper.ToUpper = "UTILITY") And _
512             (CAFE.Vehicles(vehidx).VehStructure.ToUpper = "FRAME" Or _
513             CAFE.Vehicles(vehidx).VehStructure.ToUpper = "LADDER" Or _
514             CAFE.Vehicles(vehidx).CurbWeight > 4000) Then
515             ' Do not apply to pickups or utilities with frame structure or curb weight over
     4,000 pounds.
516             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehROLL) =
    False
517         End If
518
519         ' Low-drag brakes (LDB)
520         If CAFE.Vehicles(vehidx).TechOverLDB = CAFE.TechOverRide.Used Then
521             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tVehLDB) = True
522             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehLDB) =
    False
523         ElseIf (CAFE.Vehicles(vehidx).Style.ToUpper = "PICKUP" Or _
524             CAFE.Vehicles(vehidx).Style.ToUpper.ToUpper = "UTILITY") And _
525             (CAFE.Vehicles(vehidx).VehStructure.ToUpper = "FRAME" Or _
526             CAFE.Vehicles(vehidx).VehStructure.ToUpper = "LADDER" Or _
527             CAFE.Vehicles(vehidx).CurbWeight > 4000) Then
528             ' Do not apply to pickups or utilities with frame structure or curb weight over
     4,000 pounds.
529             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehLDB) =
    False
530         End If
531
532         ' Front axle disconnect (FAD)
533         If CAFE.Vehicles(vehidx).TechOverFAD = CAFE.TechOverRide.Used Then
534             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tVehFAD) = True
535             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehFAD) =
    False
536         ElseIf CAFE.Vehicles(vehidx).Drive.ToUpper = "2" Or _
537             CAFE.Vehicles(vehidx).Drive.ToUpper = "F" Or _
538             CAFE.Vehicles(vehidx).Drive.ToUpper = "R" Then
539             ' FAD is not applicable to 2WD vehicles.
540             ' TODO:  Check to see if distinction between AWD and 4WD is needed.
```

```
541             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehFAD) =
    False
542         End If
543
544         ' Aerodynamic drag reduction (AER)
545         ' TODO:  Decide on maximum eligible Cd.
546         ' If CAFE.Vehicles(vehidx).AerodynamicDragCoefficient < Cd_max then...
547         If CAFE.Vehicles(vehidx).IsTechOverRideUsed(TechnologiesClass.TechIndexes.tVehAERO)
     Then
548             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tVehAERO) = True
549             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehAERO) =
    False
550         End If
551
552         ' Engine accessory improvement (EAI)
553         If CAFE.Vehicles(vehidx).IsTechOverRideUsed(TechnologiesClass.TechIndexes.tEngEAI)
    Then
554             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tEngEAI) = True
555             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tEngEAI) =
    False
556         End If
557
558         ' 42 Volt electrical system (42V or V42)
559         If CAFE.Vehicles(vehidx).TechOver42V = CAFE.TechOverRide.Used Then
560             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tVeh42V) = True
561             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVeh42V) =
    False
562         ElseIf CAFE.Transmissions(tranidx).TechOverHEV = CAFE.TechOverRide.Used Or _
563         CAFE.Vehicles(vehidx).HybridType.ToUpper = "E" Then
564             ' If hybrid electric vehicle, assume 42 Volt system inapplicable.
565             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVeh42V) =
    False
566         End If
567
568         ' Electric power steering (EPS)
569         ' TODO:  Check to see if EPS requires 42V system.
570         If CAFE.Vehicles(vehidx).TechOverEPS = CAFE.TechOverRide.Used Then
571             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tVehEPS) = True
572             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehEPS) =
    False
573         End If
574
575         ' Integrated starter/generator (ISG)
576         If CAFE.Vehicles(vehidx).TechOverISG = CAFE.TechOverRide.Used Then
577             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tVehISG) = True
578             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehISG) =
    False
579         ElseIf CAFE.Transmissions(tranidx).TechOverHEV = CAFE.TechOverRide.Used OrElse _
580         CAFE.Vehicles(vehidx).HybridType.ToUpper = "E" OrElse _
581         CAFE.Vehicles(vehidx).HybridType.ToUpper = "H" Then
582             ' If hybrid vehicle, assume ISG inapplicable.
583             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tVehISG) =
    False
584         End If
585
586         ' Midrange hybridization (HEV)
587         If CAFE.Transmissions(tranidx).TechOverHEV = CAFE.TechOverRide.Used OrElse _
588         CAFE.Vehicles(vehidx).HybridType.ToUpper = "E" OrElse _
589         CAFE.Vehicles(vehidx).HybridType.ToUpper = "H" Then
590             CAFE.Vehicles(vehidx).TechUsed(TechnologiesClass.TechIndexes.tTrnHEV) = True
591             CAFE.Vehicles(vehidx).TechApplies(TechnologiesClass.TechIndexes.tTrnHEV) =
    False
592         End If
593
594     End Sub ' InitTechnologyApplies
595     Private Sub SetTechPathOverride(ByVal Setting As String, ByRef TechEnab As Boolean)
596         ' SetTechPathOverride allows user to force technologies to be enabled or disabled
    on an
597         ' engine-, transmission-, or vehicle-specific basis, notwithstanding (i.e.,
    overriding)
```

```
598         ' selected technology path.  Settings include:
599         '   ALLOW:  forces the technology to be enabled
600         '   SKIP:   forces the technology to be disabled
601         ' NOTE:  The user can use the same fields to identify technologies as having been
602         ' used by specific engines, transmissions, or vehicles.  Such cases are handled separately
603         ' in the InitTechnologyApplies subroutine.
604         If Setting.ToUpper = "ALLOW" Then
605             TechEnab = True
606         ElseIf Setting.ToUpper = "SKIP" Then
607             TechEnab = False
608         End If
609     End Sub
610
611 End Class   ' Vehicle Technology Class
612
613
```

```
1  Public Module xlUtilities
2
3  #Region " Constants and Enumerators "
4
5      Public Const NewLine As String = Chr(13) + Chr(10)
6
7      Private Const DBUG As Boolean = True
8      Private ActSheet As Excel.Worksheet
9
10     Public Enum Margin As Integer
11         ' in inches
12         _0_00_INCH = 0
13         _0_25_INCH = 18
14         _0_50_INCH = 36
15         _0_75_INCH = 54
16         _1_00_INCH = 72
17         _1_25_INCH = 90
18         _1_50_INCH = 108
19         _1_75_INCH = 126
20         _2_00_INCH = 144
21         _2_25_INCH = 162
22         _3_50_INCH = 180
23         ' in pixels
24         _000_PIXELS = 0
25         _018_PIXELS = 18
26         _036_PIXELS = 36
27         _054_PIXELS = 54
28         _072_PIXELS = 72
29         _090_PIXELS = 90
30         _108_PIXELS = 108
31         _126_PIXELS = 126
32         _144_PIXELS = 144
33         _162_PIXELS = 162
34         _180_PIXELS = 180
35     End Enum
36
37 #End Region
38
39
40 #Region " Methods for writing data to excel "
41
42     '''
43     ' This method writes the specified value to the given buffer at the specified row and column.
44     '
45     ' Params:
46     '   buffer          - the Excel.Range buffer where to write the data to
47     '   val             - the value to write to the buffer
48     '   row             - the row index of the cell in the the buffer where the value will be writen
49     '   col             - the column index of the cell in the the buffer where the value will be
50     '                     writen
51     '   incrementRow    - optional; indicates whether or not to increment the row counter
52     '   incrementCol    - optional; indicates whether or not to increment the col counter
53     '
54     Public Sub WriteCell(ByRef buffer As Excel.Range, _
55                          ByRef val As Object, _
56                          ByRef row As Integer, _
57                          ByRef col As Integer, _
58                          Optional ByRef incrementRow As Boolean = False, _
59                          Optional ByRef incrementCol As Boolean = False)
60         '
61         CAFE.Profiling.FunctionStart()
62
63         '
64         ' writes one cell to buffer
65         If IsNothing(val) Then
66             buffer(row, col) = ""
67         Else
```

```
68              buffer(row, col) = val
69          End If
70
71          If incrementRow Then row += 1
72          If incrementCol Then col += 1
73
74          CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
75      End Sub
76
77      '''
78      ' This method writes the specified value, formatted as bold, to the given buffer at the
79      ' specified row and column.
80      '
81      ' Params:
82      '   buffer          - the Excel.Range buffer where to write the data to
83      '   val             - the value to write to the buffer
84      '   row             - the row index of the cell in the the buffer where the value will be writen
85      '   col             - the column index of the cell in the the buffer where the value will be
86      '                     writen
87      '   incrementRow    - optional; indicates whether or not to increment the row counter
88      '   incrementCol    - optional; indicates whether or not to increment the col counter
89      '
90      Public Sub WriteBoldCell(ByRef buffer As Excel.Range, _
91                               ByRef val As Object, _
92                               ByRef row As Integer, _
93                               ByRef col As Integer, _
94                               Optional ByRef incrementRow As Boolean = False, _
95                               Optional ByRef incrementCol As Boolean = False)
96          '
97          CAFE.Profiling.FunctionStart()
98
99          '
100         ' writes a cell formatted as bold to buffer
101         SetFontBold(buffer(row, col))
102         WriteCell(buffer, val, row, col, incrementRow, incrementCol)
103
104         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
105     End Sub
106
107     '''
108     ' This method writes the specified value, formatted with indents, to the given buffer at the
109     ' specified row and column.
110     '
111     ' Params:
112     '   buffer          - the Excel.Range buffer where to write the data to
113     '   val             - the value to write to the buffer
114     '   indentation     - the amount by which to indent the data in the cell specified by the given
115     '                     row and column
116     '   row             - the row index of the cell in the the buffer where the value will be writen
117     '   col             - the column index of the cell in the the buffer where the value will be
118     '                     writen
119     '   incrementRow    - optional; indicates whether or not to increment the row counter
120     '   incrementCol    - optional; indicates whether or not to increment the col counter
121     '
122     Public Sub WriteIndentedCell(ByRef buffer As Excel.Range, _
123                                  ByRef val As Object, _
124                                  ByRef indentation As Integer, _
125                                  ByRef row As Integer, _
126                                  ByRef col As Integer, _
127                                  Optional ByRef incrementRow As Boolean = False, _
128                                  Optional ByRef incrementCol As Boolean = False)
129         '
130         CAFE.Profiling.FunctionStart()
```

```
131
132         '
133         ' writes a cell formatted with indents to buffer
134         SetIndent(buffer(row, col), indentation)
135         WriteCell(buffer, val, row, col, incrementRow, incrementCol)
136
137         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
138     End Sub
139
140     '''
141     ' This method writes the specified value, formatted as a number, to the given buffer at the
142     ' specified row and column.
143     '
144     ' Params:
145     '   buffer          - the Excel.Range buffer where to write the data to
146     '   val             - the value to write to the buffer
147     '   numberFormat    - the number format to apply to the cell specified by the given row and col
148     '   row             - the row index of the cell in the the buffer where the value will be writen
149     '   col             - the column index of the cell in the the buffer where the value will be
150     '                     writen
151     '   incrementRow    - optional; indicates whether or not to increment the row counter
152     '   incrementCol    - optional; indicates whether or not to increment the col counter
153     '
154     Public Sub WriteNumberCell(ByRef buffer As Excel.Range, _
155                                ByRef val As Object, _
156                                ByRef numberFormat As String, _
157                                ByRef row As Integer, _
158                                ByRef col As Integer, _
159                                Optional ByRef incrementRow As Boolean = False, _
160                                Optional ByRef incrementCol As Boolean = False)
161         '
162         CAFE.Profiling.FunctionStart()
163
164         '
165         ' writes a cell with the given number format to buffer
166         SetNumberFormat(buffer(row, col), numberFormat)
167         WriteCell(buffer, val, row, col, incrementRow, incrementCol)
168
169         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
170     End Sub
171
172     '''
173     ' This method writes the specified value, formatted as text, to the given buffer at the
174     ' specified row and column.
175     '
176     ' Params:
177     '   buffer          - the Excel.Range buffer where to write the data to
178     '   val             - the value to write to the buffer
179     '   row             - the row index of the cell in the the buffer where the value will be writen
180     '   col             - the column index of the cell in the the buffer where the value will be
181     '                     writen
182     '   incrementRow    - optional; indicates whether or not to increment the row counter
183     '   incrementCol    - optional; indicates whether or not to increment the col counter
184     '
185     Public Sub WriteTextCell(ByRef buffer As Excel.Range, _
186                              ByRef val As Object, _
187                              ByRef row As Integer, _
188                              ByRef col As Integer, _
189                              Optional ByRef incrementRow As Boolean = False, _
190                              Optional ByRef incrementCol As Boolean = False)
191         '
192         CAFE.Profiling.FunctionStart()
193
```

```
194         '
195         ' writes a cell formatted as a text to buffer
196         WriteNumberCell(buffer, val, "@", row, col, incrementRow, incrementCol)
197
198         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
199     End Sub
200
201 #End Region
202
203
204 #Region " Page setup, sorting, and header/footer setup "
205
206     '''
207     ' This method should not be used, but keep it around just in case.
208     '
209     Private Sub SetWorksheetHeaderAndFooter(ByRef HeaderText As String, _
210                                             Optional ByRef NarrowMargins As Boolean =
    False)
211         '
212         CAFE.Profiling.FunctionStart()
213
214         '
215         ' Sets the header and footer.
216         ' Also set to be 1 page wide by whatever in Landscape mode
217         '
218         ActSheet = ExcelApp.ActiveSheet
219         If NarrowMargins Then
220             ActSheet.PageSetup.LeftMargin = 36    ' 36 is half an inch.
221             ActSheet.PageSetup.RightMargin = 36   ' There used to be a conversion routine
    in VB6
222         End If
223         SetWorksheetHeader(HeaderText)
224         SetWorksheetFooter()
225
226         ActSheet.PageSetup.Orientation = Excel.XlPageOrientation.xlLandscape
227         ActSheet.PageSetup.Zoom = False
228
229         ActSheet.PageSetup.FitToPagesWide = 1
230         ActSheet.PageSetup.FitToPagesTall = False
231
232
233         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
234     End Sub
235
236     '''
237     ' This method sets up the active worksheet for printing.
238     '
239     ' Params:
240     '   Oreintation     - optional; specifies whether to use portrait or landscape
    orientation;
241     '                     default is portrait
242     '   LeftMargin      - optional; specifies what the left margin's length should be
243     '   RightMargin     - optional; specifies what the right margin's length should be
244     '   TopMargin       - optional; specifies what the top margin's length should be
245     '   BottomMargin    - optional; specifies what the bottom margin's length should be
246     '   FitToPagesWide  - optional; specifies the maximum number of pages in width that
    the data
247     '                     should occupy
248     '   FitToPagesTall  - optional; specifies the maximum number of pages in height that
    the data
249     '                     should occupy
250     '
251     Public Sub PageSetup(Optional ByRef Oreintation As Excel.XlPageOrientation = Excel.
    XlPageOrientation.xlPortrait, _
252                          Optional ByRef LeftMargin As Margin = Margin._0_75_INCH, _
253                          Optional ByRef RightMargin As Margin = Margin._0_75_INCH, _
254                          Optional ByRef TopMargin As Margin = Margin._1_00_INCH, _
255                          Optional ByRef BottomMargin As Margin = Margin._1_00_INCH, _
256                          Optional ByRef FitToPagesWide As Integer = -1, _
257                          Optional ByRef FitToPagesTall As Integer = -1)
```

```
258         '
259         CAFE.Profiling.FunctionStart()
260
261         '
262         ' Sets up the excel worksheet for display
263         '
264         ActSheet = ExcelApp.ActiveSheet
265         ' set orientation and zoom
266         ActSheet.PageSetup.Orientation = Excel.XlPageOrientation.xlLandscape
267         ActSheet.PageSetup.Zoom = False
268         ' set margins
269         ActSheet.PageSetup.LeftMargin = LeftMargin
270         ActSheet.PageSetup.RightMargin = RightMargin
271         ActSheet.PageSetup.TopMargin = TopMargin
272         ActSheet.PageSetup.BottomMargin = BottomMargin
273         ActSheet.PageSetup.HeaderMargin = Math.Max(0, TopMargin - 18)
274         ActSheet.PageSetup.FooterMargin = Math.Max(0, BottomMargin - 18)
275         ' set scaling
276         If FitToPagesTall <= 0 Then
277             ActSheet.PageSetup.FitToPagesTall = False
278         Else
279             ActSheet.PageSetup.FitToPagesTall = FitToPagesTall
280         End If
281         If FitToPagesWide <= 0 Then
282             ActSheet.PageSetup.FitToPagesWide = False
283         Else
284             ActSheet.PageSetup.FitToPagesWide = FitToPagesWide
285         End If
286
287         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
288     End Sub
289
290     '''
291     ' This method sets up the active worksheet's header, adding the specified HeaderText ↙
        and the
292     ' current data and time.
293     '
294     ' Params:
295     '   HeaderText      - the title of the worksheet to display in the worksheet's header.
296     '
297     Public Sub SetWorksheetHeader(ByRef HeaderText As String)
298         CAFE.Profiling.FunctionStart()
299
300         ' Sets up the header of the excel worksheet
301         '
302         ActSheet = ExcelApp.ActiveSheet
303
304         With ActSheet.PageSetup
305             ' set text and date
306             If HeaderText <> "" Then
307                 .CenterHeader = "&B" & HeaderText
308             End If
309             .RightHeader = "&08 " & Now.ToString
310         End With
311
312         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
313     End Sub
314
315     '''
316     ' This method sets up the active worksheet's footer, adding a "CONFIDENTIAL" warning, ↙
        among
317     ' other things.
318     '
319     Public Sub SetWorksheetFooter()
320         CAFE.Profiling.FunctionStart()
321
322         ' Sets up the header of the excel worksheet
323         '
324         ActSheet = ExcelApp.ActiveSheet
325
```

```
326         ActSheet.PageSetup.LeftFooter = "&08&F / &A"
327         ActSheet.PageSetup.CenterFooter = "&" & Chr(34) & "Arial,Bold" & Chr(34) & "&
    12CONFIDENTIAL -- CONTAINS PROTECTED MANUFACTURER DATA"
328         ActSheet.PageSetup.RightFooter = "&08Page &P of &N"
329
330         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
331     End Sub
332
333     '''
334     ' This method inserts a page break at the specified cell.
335     '
336     Public Sub InsertPageBreak(ByRef cell As Object)
337         CAFE.Profiling.FunctionStart()
338
339         Try
340             ExcelApp.ActiveSheet.HPageBreaks.Add(cell)
341             'ActSheet.HPageBreaks.Add(cell)
342         Catch ex As Exception
343             If DBUG Then
344                 MsgBox("In InsertPageBreak(cell): " & ex.Message)
345             End If
346
347             MsgBox("Incorrect cell specified; page break not added.")
348         End Try
349
350         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
351     End Sub
352
353     '''
354     ' This method freezes the panes around the specified cell within the given range.
355     '
356     ' Params:
357     '   range           - the pointer to the range object on the active sheet
358     '   cell            - the cell around which to freeze the panes
359     '
360     Public Sub FreezePanes(ByRef range As Excel.Range, ByRef cell As Object)
361         CAFE.Profiling.FunctionStart()
362
363         Try
364             If Not IsNothing(cell) AndAlso cell <> String.Empty Then
365                 range.Range(cell).Select()
366                 ExcelApp.ActiveWindow.FreezePanes = True
367             End If
368         Catch ex As Exception
369             If DBUG Then
370                 MsgBox("In FreezePanes(range, cell): " & ex.Message)
371             End If
372         End Try
373
374         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
375     End Sub
376
377     '''
378     ' This method sets up the worksheet to display title rows when printed.  Title rows
    are
379     ' displayed on every printed page at the top.
380     '
381     ' Params:
382     '   startingRow     - the first title row to display on every page
383     '   endingRow       - the last title row to display on every page
384     '
385     Public Sub PrintTitleRows(ByRef startingRow As Integer, ByRef endingRow As Integer)
386         CAFE.Profiling.FunctionStart()
387
388         PrintTitleRowsAndCols(startingRow, endingRow, -1, -1)
389
390         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
391     End Sub
392
393     '''
```

```
394     ' This method sets up the worksheet to display title columns when printed.  Title
        columns are
395     ' displayed on every printed page on the left.
396     '
397     ' Params:
398     '   startingCol     - the first title column to display on every page
399     '   endingCol       - the last title column to display on every page
400     '
401     Public Sub PrintTitleCols(ByRef startingCol As Integer, ByRef endingCol As Integer)
402         CAFE.Profiling.FunctionStart()
403
404         PrintTitleRowsAndCols(-1, -1, startingCol, endingCol)
405
406         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
407     End Sub
408
409     '''
410     ' This method sets up the worksheet to display title rows and title columns when
        printed.  Title
411     ' rows and columns are displayed on every printed page, either at the top, for rows,
        or on the
412     ' left, for columns.
413     '
414     ' Params:
415     '   r1              - the first title row to display on every page
416     '   r2              - the last title row to display on every page
417     '   c1              - the first title column to display on every page
418     '   c2              - the last title column to display on every page
419     '
420     Private Sub PrintTitleRowsAndCols(ByRef r1 As Integer, ByRef r2 As Integer, _
421                                       ByRef c1 As Integer, ByRef c2 As Integer)
422         '
423         CAFE.Profiling.FunctionStart()
424
425         '
426         ActSheet = ExcelApp.ActiveSheet
427
428         If r1 > 0 AndAlso r2 > 0 Then
429             ActSheet.PageSetup.PrintTitleRows = "$" & r1.ToString & ":$" & r2.ToString
430         End If
431         '
432         If c1 > 0 AndAlso c2 > 0 Then
433             ActSheet.PageSetup.PrintTitleColumns = "$" & c1.ToString & ":$" & c2.ToString
434         End If
435
436         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
437     End Sub
438
439     '''
440     ' This method sorts the data within the given range according to the specified keys
        and sort
441     ' orders.
442     '
443     ' Params:
444     '   range           - the range within which to sort the data
445     '   key1            - the cell around which the primary sort of the data is to be
        performed
446     '   order1          - optional; specifies the descending or ascending order for the
        primary sort
447     '   key2            - optional; the cell around which the second sort of the data is
        to be
448     '                     performed
449     '   order2          - optional; specifies the descending or ascending order for the
        second sort
450     '   key3            - optional; the cell around which the third sort of the data is to
         be
451     '                     performed
452     '   order3          - optional; specifies the descending or ascending order for the
        third sort
453     '   sortOrientation - optional; specifies whether to sort rows or columns within the
```

```
    given range
454     '
455     Public Sub SortData(ByRef range As Excel.Range, _
456                         ByRef key1 As Object, _
457                         Optional ByRef order1 As Excel.XlSortOrder = Excel.XlSortOrder.
    xlAscending, _
458                         Optional ByRef key2 As Object = Nothing, _
459                         Optional ByRef order2 As Excel.XlSortOrder = Excel.XlSortOrder.
    xlAscending, _
460                         Optional ByRef key3 As Object = Nothing, _
461                         Optional ByRef order3 As Excel.XlSortOrder = Excel.XlSortOrder.
    xlAscending, _
462                         Optional ByRef sortOrientation As Excel.XlSortOrientation = Excel.
    XlSortOrientation.xlSortRows)
463         '
464         CAFE.Profiling.FunctionStart()
465
466         '
467         Try
468             Dim skey1, skey2, skey3 As Object
469
470             If Not IsNothing(key1) Then skey1 = range.Range(key1)
471             If Not IsNothing(key2) Then skey2 = range.Range(key2)
472             If Not IsNothing(key3) Then skey3 = range.Range(key3)
473
474
475             range.Sort(skey1, order1, skey2, , order2, skey3, order3; _
476                        Excel.XlYesNoGuess.xlNo, 1, False, sortOrientation, , Excel.
    XlSortDataOption.xlSortNormal, _
477                        Excel.XlSortDataOption.xlSortNormal, Excel.XlSortDataOption.
    xlSortNormal)
478         Catch ex As Exception
479             If DBUG Then
480                 MsgBox("In SortData(range, key1, optional order1, optional key2, optional
    order2, " & _
481                        "optional key3, optional order3, optional sortOrientation): " & ex.
    Message)
482             End If
483         End Try
484
485         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
486     End Sub
487
488 #End Region
489
490
491 #Region " Borders "
492
493     '''
494     ' This method sets all cell borders within the given range.  By default, the cell
    borders will
495     ' be set up as black, thin, and continuous.
496     '
497     ' Params:
498     '   range           - the range within which to set the border
499     '   style           - optional; specifies the line style to apply to the cell borders
    within the
500     '                     given range
501     '   weight          - optional; specifies the border weight to apply to the cell
    borders within
502     '                     the given range
503     '   color           - optional; specifies the color that the cell borders within the
    given range
504     '                     should be
505     '
506     Public Sub SetBorders(ByRef range As Excel.Range, _
507                           Optional ByRef style As Excel.XlLineStyle = Excel.XlLineStyle.
    xlContinuous, _
508                           Optional ByRef weight As Excel.XlBorderWeight = Excel.
    XlBorderWeight.xlThin, _
```

```
509                         Optional ByRef color As Object = "Black")
510         '
511         CAFE.Profiling.FunctionStart()
512
513         '
514         ' Sets cell borders
515         SetBorder(range, Excel.XlBordersIndex.xlEdgeLeft, style, weight, color)
516         SetBorder(range, Excel.XlBordersIndex.xlEdgeRight, style, weight, color)
517         SetBorder(range, Excel.XlBordersIndex.xlEdgeTop, style, weight, color)
518         SetBorder(range, Excel.XlBordersIndex.xlEdgeBottom, style, weight, color)
519         SetBorder(range, Excel.XlBordersIndex.xlInsideHorizontal, style, weight, color)
520         SetBorder(range, Excel.XlBordersIndex.xlInsideVertical, style, weight, color)
521
522         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
523     End Sub
524
525     '''
526     ' This method sets a single border index within the given range.  By default, the specified
527     ' border will be set up as black, thin, and continuous.
528     '
529     ' Params:
530     '   range           - the range within which to set the border
531     '   border          - the border index to apply to all cells within the given range; the border
532     '                     indexes are defined as follows: xlEdgeLeft, xlEdgeRight, xlEdgeTop,
533     '                     xlEdgeBottom, xlInsideHorizontal, xlInsideVertical, xlDiagonalDown, and
534     '                     xlDiagonalUp
535     '   style           - optional; specifies the line style to apply to the specified border within
536     '                     the given range
537     '   weight          - optional; specifies the border weight to apply to the specified border
538     '                     within the given range
539     '   color           - optional; specifies the color that the specified border within the given
540     '                     range should be
541     '
542     Public Sub SetBorder(ByRef range As Excel.Range, _
543                          ByRef border As Excel.XlBordersIndex, _
544                          Optional ByRef style As Excel.XlLineStyle = Excel.XlLineStyle.xlContinuous, _
545                          Optional ByRef weight As Excel.XlBorderWeight = Excel.XlBorderWeight.xlThin, _
546                          Optional ByRef color As Object = "Black")
547         '
548         CAFE.Profiling.FunctionStart()
549
550         '
551         ' Sets a single border
552         '
553         Try
554             Dim MyColor As System.Drawing.Color = GetColor(color)
555             With range.Borders(border)
556                 .LineStyle = style
557                 .Weight = weight
558                 .Color = RGB(MyColor.R, MyColor.G, MyColor.B)  ' CType(borderColor, Color)
559             End With
560         Catch ex As Exception
561             If DBUG Then
562                 'MsgBox("In SetBorder(range, border, optional style, " & _
563                 '       "optional weight, optional color): " & ex.Message)
564             End If
565         End Try
566
567         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
568     End Sub
```

```
569
570 #End Region
571
572
573 #Region " Font "
574
575     '''
576     ' This method sets and formats the font within the given range.  By default, if SetFont is
577     ' called with no arguments other than range, the range will be formatted with Arial 10 pt. Black
578     ' font, with 0 indentation.
579     '
580     ' Params:
581     '   range           - the range to which to apply the font
582     '   name            - optional; the name of the font to apply to the given range; default font
583     '                     is "Arial"
584     '   bold            - optional; specifies whether the data within the given range should be bold
585     '   italic          - optional; specifies whether the data within the given range should be italic
586     '   underline       - optional; specifies whether to underline the data within the given range
587     '   size            - optional; specifies the size of the font within the given range
588     '   color           - optional; specifies the color of the font within the given range
589     '   indent          - optional; specifies the amount of indentation to apply to the text within
590     '                     the given range; if indent is <= 0, then no indentation is applied
591     '
592     Public Sub SetFont(ByRef range As Excel.Range, _
593                        Optional ByRef name As String = "Arial", _
594                        Optional ByRef bold As Boolean = False, _
595                        Optional ByRef italic As Boolean = False, _
596                        Optional ByRef underline As Boolean = False, _
597                        Optional ByRef size As Integer = 10, _
598                        Optional ByRef color As Object = "Black", _
599                        Optional ByRef indent As Integer = 0)
600         '
601         CAFE.Profiling.FunctionStart()
602
603         '
604         ' Sets the font
605         range.Font.Name = name
606         range.Font.Bold = bold
607         range.Font.Italic = italic
608         range.Font.Underline = underline
609         range.Font.Size = size
610         SetFontColor(range, GetColor(color))
611         SetIndent(range, indent)
612
613         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
614     End Sub
615
616     '''
617     ' This method bolds or un-bolds the font within the given range.
618     '
619     ' Params:
620     '   range           - the range to bold/un-bold
621     '   bold            - optional; specifies whether the data within the given range should be
622     '                     bolded; by default, the data will be bolded
623     '
624     Public Sub SetFontBold(ByRef range As Excel.Range, Optional ByRef bold As Boolean = True)
625         CAFE.Profiling.FunctionStart()
626
627         ' Bolds the text
628         range.Font.Bold = bold
```

```
629
630         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
631     End Sub
632
633     '''
634     ' This method italicizes or "un"-italicizes the font within the given range.
635     '
636     ' Params:
637     '   range           - the range to italicize/"un"-italicize
638     '   italic          - optional; specifies whether the data within the given range should be
639     '                     italicized; by default, the data will be italicized
640     '
641     Public Sub SetFontItalic(ByRef range As Excel.Range, Optional ByRef italic As Boolean = True)
642         CAFE.Profiling.FunctionStart()
643
644         ' Makes text italic
645         range.Font.Italic = italic
646
647         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
648     End Sub
649
650     '''
651     ' This method underlines or "un"-underlines the font within the given range.
652     '
653     ' Params:
654     '   range           - the range to underline/"un"-underline
655     '   underline       - optional; specifies whether the data within the given range should be
656     '                     underlined; by default, the data will be underlined
657     '
658     Public Sub SetFontUnderline(ByRef range As Excel.Range, Optional ByRef underline As Boolean = True)
659         CAFE.Profiling.FunctionStart()
660
661         ' Underline the text
662         range.Font.Underline = underline
663
664         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
665     End Sub
666
667     '''
668     ' This method sets the size of the font within the given range.
669     '
670     ' Params:
671     '   range           - the range within which to set the size of the font
672     '   size            - the size to apply to the font within the given range
673     '
674     Public Sub SetFontSize(ByRef range As Excel.Range, ByRef size As Integer)
675         CAFE.Profiling.FunctionStart()
676
677         ' Sets font size
678         range.Font.Size = size
679
680         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
681     End Sub
682
683     '''
684     ' This method sets the font's color within the given range.
685     '
686     ' Params:
687     '   range           - the range within which to set the font's color
688     '   color           - the color to apply to the font within the given range
689     '
690     Public Sub SetFontColor(ByRef range As Excel.Range, ByRef color As Color)
691         CAFE.Profiling.FunctionStart()
692
693         ' Sets font color
694         range.Font.Color = RGB(color.R, color.G, color.B)
```

```
695
696         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
697     End Sub
698
699     '''
700     ' This method indents the text by the specified amount within the specified range.
701     '
702     ' Params:
703     '   range           - the range within which to indent the text
704     '   indent          - the amount of indentation to apply to the text within the given range; if
705     '                     indent is <= 0, then no indentation is applied
706     '
707     Public Sub SetIndent(ByRef range As Excel.Range, ByRef indent As Integer)
708         CAFE.Profiling.FunctionStart()
709
710         ' Sets the indentation
711         If indent > 0 Then
712             range.IndentLevel = indent
713         End If
714
715         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
716     End Sub
717
718 #End Region
719
720
721 #Region " General cell formating "
722
723     '''
724     ' This method merges (and optionally centers) the data within the given range.
725     '
726     ' Params:
727     '   range           - the range of data to merge
728     '   andCenter       - optional; determines whether the merged data should also be centered
729     '                     horizontally
730     '   andVCenter      - optional; determines whether the merged data should also be centered
731     '                     vertically
732     '
733     Public Sub Merge(ByRef range As Excel.Range, _
734                      Optional ByRef andCenter As Boolean = False, _
735                      Optional ByRef andVCenter As Boolean = False)
736         '
737         CAFE.Profiling.FunctionStart()
738
739         '
740         range.Merge()
741         If andCenter Then
742             AlignText(range, Excel.XlHAlign.xlHAlignCenter)
743         End If
744         If andVCenter Then
745             AlignText(range, Excel.XlVAlign.xlVAlignCenter)
746         End If
747
748         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
749     End Sub
750
751     '''
752     ' This method merges (and optionally centers) the data within the given range between cell1 and
753     ' cell2.
754     '
755     ' Params:
756     '   range           - the pointer to the range object on the active sheet
757     '   cell1           - cell1 and cell2 are the cells between which to merge the data
758     '   cell2           - cell1 and cell2 are the cells between which to merge the data
759     '   andCenter       - optional; determines whether the merged data should also be centered
```

```
760     '                       horizontally
761     '   andVCenter      - optional; determines whether the merged data should also be
        centered
762     '                       vertically
763     '
764     Public Sub Merge(ByRef range As Excel.Range, _
765                      ByRef cell1 As Object, _
766                      ByRef cell2 As Object, _
767                      Optional ByRef andCenter As Boolean = False, _
768                      Optional ByRef andVCenter As Boolean = False)
769         '
770         CAFE.Profiling.FunctionStart()
771
772         '
773         Try
774             xlUtilities.Merge(range.Range(cell1, cell2), andCenter, andVCenter)
775         Catch ex As Exception
776             If DBUG Then
777                 MsgBox("In Merge(range, cell1, cell2, optional andCenter): " & ex.Message)
778             End If
779         End Try
780
781         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
782     End Sub
783
784     '''
785     ' This method merges (and optionally centers) the data within the given range
        specified by the
786     ' given coordinates.
787     '
788     ' Params:
789     '   range           - the pointer to the range object on the active sheet
790     '   x1, y1, x2, y2  - the coordinates of the first and second cells
791     '   andCenter       - optional; determines whether the merged data should also be
        centered
792     '                     horizontally
793     '   andVCenter      - optional; determines whether the merged data should also be
        centered
794     '                     vertically
795     '
796     Public Sub Merge(ByRef range As Excel.Range, _
797                      ByRef x1 As Integer, _
798                      ByRef y1 As Integer, _
799                      ByRef x2 As Integer, _
800                      ByRef y2 As Integer, _
801                      Optional ByRef andCenter As Boolean = False, _
802                      Optional ByRef andVCenter As Boolean = False)
803         '
804         CAFE.Profiling.FunctionStart()
805
806         '
807         Dim cell1 As Object = range.Cells(x1, y1)
808         Dim cell2 As Object = range.Cells(x2, y2)
809
810         xlUtilities.Merge(range, cell1, cell2, andCenter, andVCenter)
811
812         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
813     End Sub
814
815     '''
816     ' This method sets the number format withing the given range.
817     '
818     ' Params:
819     '   range           - the range within which to apply the number format
820     '   format          - the number format to apply to the given range
821     '
822     Public Sub SetNumberFormat(ByRef range As Excel.Range, ByRef format As String)
823         CAFE.Profiling.FunctionStart()
824
825         range.NumberFormat = format
```

```
826
827         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
828     End Sub
829
830     '''
831     ' This method aligns a all data horizontally within the given range.
832     '
833     ' Params:
834     '   range           - the range within which to align the data
835     '   hAlign          - the horizontal alignment to apply to the entire worksheet
836     '
837     Public Sub AlignText(ByRef range As Excel.Range, ByRef hAlign As Excel.XlHAlign)
838         CAFE.Profiling.FunctionStart()
839
840         range.HorizontalAlignment = hAlign
841
842         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
843     End Sub
844
845     '''
846     ' This method aligns a ali data vertically within the given range.
847     '
848     ' Params:
849     '   range           - the range within which to align the data
850     '   vAlign          - the vertical alignment to apply to the entire worksheet
851     '
852     Public Sub AlignText(ByRef range As Excel.Range, ByRef vAlign As Excel.XlVAlign)
853         CAFE.Profiling.FunctionStart()
854
855         range.VerticalAlignment = vAlign
856
857         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
858     End Sub
859
860     '''
861     ' This method aligns a all data horizontally and vertically within the given range.
862     '
863     ' Params:
864     '   range           - the range within which to align the data
865     '   hAlign          - the horizontal alignment to apply to the entire worksheet
866     '   vAlign          - the vertical alignment to apply to the entire worksheet
867     '
868     Public Sub AlignText(ByRef range As Excel.Range, _
869                          ByRef hAlign As Excel.XlHAlign, _
870                          ByRef vAlign As Excel.XlVAlign)
871         '
872         CAFE.Profiling.FunctionStart()
873
874         '
875         range.HorizontalAlignment = hAlign
876         range.VerticalAlignment = vAlign
877
878         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
879     End Sub
880
881     '''
882     ' This method aligns all data from cell "A1" to cell "IV65536" horizontally and ↙
    vertically.
883     '
884     ' Params:
885     '   range           - the pointer to the range object on the active sheet
886     '   hAlign          - the horizontal alignment to apply to the entire worksheet
887     '   vAlign          - optional; the vertical alignment to apply to the entire ↙
    worksheet
888     '
889     Public Sub AlignAllText(ByRef range As Excel.Range, _
890                             ByRef hAlign As Excel.XlHAlign, _
891                             Optional ByRef vAlign As Excel.XlVAlign = Excel.XlVAlign. ↙
    xlVAlignBottom)
892         '
```

```
893         CAFE.Profiling.FunctionStart()
894
895         '
896         ' alligns the entire worksheet
897         ' a better way is to set the Excel.Range in memory to the
898         ' entire worksheet, and then call AlignText(range, ,) method
899         '
900         range.Range("A1:IV65536").HorizontalAlignment = hAlign
901         range.Range("A1:IV65536").VerticalAlignment = vAlign
902
903         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
904     End Sub
905
906     '''
907     ' This method auto-fits all data within a given range.
908     '
909     ' Params:
910     '   range           - the range within which to auto-fit all data
911     '
912     Public Sub AutoFit(ByRef range As Excel.Range)
913         CAFE.Profiling.FunctionStart()
914
915         Try
916             'range.Columns.WrapText = False
917             'range.ColumnWidth = 72
918             'range.RowHeight = 72
919
920             range.Rows.AutoFit()
921             range.Columns.AutoFit()
922         Catch ex As Exception
923             If DBUG Then
924                 MsgBox("In AutoFit(range): " & ex.Message)
925             End If
926         End Try
927
928         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
929     End Sub
930
931     '''
932     ' This method auto-fits all data within a range specified by the given parameters.
933     '
934     ' Params:
935     '   range           - the pointer to the range object on the active sheet
936     '   x1              - the top-most bound of the range
937     '   y1              - the left-most bound of the range
938     '   x2              - the lowest bound of the range
939     '   y2              - the right-most bound of the range
940     '
941     Public Sub AutoFit(ByRef range As Excel.Range, _
942                        ByRef x1 As Integer, _
943                        ByRef y1 As Integer, _
944                        ByRef x2 As Integer, _
945                        ByRef y2 As Integer)
946         '
947         CAFE.Profiling.FunctionStart()
948
949         '
950         Dim cell1 = range.Cells(x1, y1)
951         Dim cell2 = range.Cells(x2, y2)
952         AutoFit(range.Range(cell1, cell2))
953
954         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
955     End Sub
956
957     '''
958     ' This method highlights all data within the given range.
959     '
960     ' Params:
961     '   range           - the range to highlight
962     '   colorIndex      - the color index to use for highlighting
```

```
963     '   pattern         - optional; the pattern to use for highlighting; default is
        xlSolid
964     '
965     Public Sub HighlightCells(ByRef range As Excel.Range, _
966                               ByRef colorIndex As Integer, _
967                               Optional ByRef pattern As Excel.Constants = Excel.Constants.
        xlSolid)
968         '
969         CAFE.Profiling.FunctionStart()
970
971         '
972         Try
973             range.Interior.ColorIndex = colorIndex
974             range.Interior.Pattern = pattern
975         Catch ex As Exception
976             If DBUG Then
977                 MsgBox("In HighlightCells(range, colorIndex, optional pattern): " & ex.
        Message)
978             End If
979         End Try
980
981         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
982     End Sub
983
984     '''
985     ' This method removes the specified row from the given range.
986     '
987     ' Params:
988     '   range           - the range to remove the row from
989     '   row             - the row index to remove
990     '
991     Public Sub RemoveRow(ByRef range As Excel.Range, ByRef row As Integer)
992         CAFE.Profiling.FunctionStart()
993
994         Dim rows As String = row.ToString() + ":" + row.ToString()
995         RemoveRows(range, rows)
996
997         CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
998     End Sub
999
1000    '''
1001    ' This method removes the specified row or rows from the given range.
1002    '
1003    ' Params:
1004    '   range           - the range to remove the rows from
1005    '   rows            - the row or rows to remove, using the following format: "12:15"
1006    '
1007    Public Sub RemoveRows(ByRef range As Excel.Range, ByRef rows As String)
1008        CAFE.Profiling.FunctionStart()
1009
1010        Try
1011            CType(range.Rows(rows), Excel.Range).EntireRow.Delete()     '(Excel.
        XlDirection.xlUp)
1012        Catch ex As Exception
1013            If DBUG Then
1014                MsgBox("In RemoveRows(range, rows): " & ex.Message)
1015            End If
1016        End Try
1017
1018        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1019    End Sub
1020
1021    '''
1022    ' This method removes the specified rows from the given range.
1023    '
1024    ' Params:
1025    '   range           - the range to remove the row from
1026    '   sRow            - the index of the starting row to remove
1027    '   eRow            - the index of the last row to remove
1028    '
```

```
1029      Public Sub RemoveRows(ByRef range As Excel.Range, ByRef sRow As Integer, ByRef eRow As ↙
           Integer)
1030          CAFE.Profiling.FunctionStart()
1031
1032          Dim rows As String = sRow.ToString() + ":" + eRow.ToString()
1033          RemoveRows(range, rows)
1034
1035          CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1036      End Sub
1037
1038      '''
1039      ' This method removes the specified column or columns from the given range.
1040      '
1041      ' Params:
1042      '   range           - the range to remove the columns from
1043      '   rows            - the column or columns to remove, using the following format: "C: ↙
          F"
1044      '
1045      Public Sub RemoveCols(ByRef range As Excel.Range, ByRef cols As String)
1046          CAFE.Profiling.FunctionStart()
1047
1048          Dim arr() As String = cols.Split(":")
1049          Dim c1 As Object = arr(0)
1050          Dim c2 As Object
1051          Dim myCols As String
1052
1053          If cols.IndexOf(":") = -1 Then
1054              ' a single col was specified
1055              c2 = arr(0)
1056          Else
1057              ' 2 cols were specified
1058              c2 = arr(1)
1059          End If
1060
1061          myCols = ConvertExcelColumnIndex(c1) + ":" + ConvertExcelColumnIndex(c2)
1062
1063          Try
1064              CType(range.Columns(myCols), Excel.Range).EntireColumn.Delete()      '(Excel.   ↙
          XlDirection.xlToLeft)
1065          Catch ex As Exception
1066              If DBUG Then
1067                  MsgBox("In RemoveCols(range, cols): " & ex.Message)
1068              End If
1069          End Try
1070
1071          CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1072      End Sub
1073
1074      '''
1075      ' This method removes the specified columns from the given range.
1076      '
1077      ' Params:
1078      '   range           - the range to remove the columns from
1079      '   sRow            - the index of the starting column to remove
1080      '   eRow            - the index of the last column to remove
1081      '
1082      Public Sub RemoveCols(ByRef range As Excel.Range, ByRef sCol As Integer, ByVal eCol As ↙
           Integer)
1083          CAFE.Profiling.FunctionStart()
1084
1085          Dim cols As String = sCol.ToString() + ":" + eCol.ToString()
1086          RemoveCols(range, cols)
1087
1088          CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1089      End Sub
1090
1091  #End Region
1092
1093
1094  #Region " Methods for naming and adding worksheets "
```

```
1095
1096        '''
1097        ' This method adds a new worksheet with the given name at the last index.
1098        '
1099        ' Params:
1100        '   name            - the name to assign to the new worksheet; the name is assigned by calling
1101        '                     the NameWorkSheet() method
1102        '
1103        Public Sub AddWorkSheet(ByRef name As String)
1104            CAFE.Profiling.FunctionStart()
1105
1106            AddWorkSheet(ExcelApp.ActiveWorkbook.Worksheets.Count + 1, name)
1107
1108            CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1109        End Sub
1110
1111        '''
1112        ' This method adds a worksheet with a given name at the given index.
1113        '
1114        ' Params:
1115        '   index           - the index at which to add the new worksheet
1116        '   name            - the name to assign to the new worksheet; the name is assigned by calling
1117        '                     the NameWorkSheet() method
1118        '
1119        Public Sub AddWorkSheet(ByRef index As Integer, ByRef name As String)
1120            CAFE.Profiling.FunctionStart()
1121
1122            Dim Wb As Excel.Workbook = ExcelApp.ActiveWorkbook
1123            Dim Ws As Excel.Sheets = Wb.Worksheets
1124
1125            If index <= 1 Then
1126                ' before first sheet
1127                Wb.Sheets.Add(Before:=Ws(1))
1128                index = 1
1129            ElseIf index > Ws.Count Then
1130                ' at the end of last sheet
1131                Wb.Sheets.Add(After:=Ws(Ws.Count))
1132                index = Ws.Count
1133            Else
1134                ' before the sheet at given index
1135                Wb.Sheets.Add(Before:=Ws(index))
1136            End If
1137
1138            NameWorkSheet(index, name)
1139
1140            CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1141        End Sub
1142
1143        '''
1144        ' This method removes a worksheet at the specified index.
1145        '
1146        ' Params:
1147        '   index           - the index of the worksheet to remove
1148        '
1149        Public Sub RemoveWorksheet(ByRef index As Integer)
1150            CAFE.Profiling.FunctionStart()
1151
1152            Dim Wb As Excel.Workbook = ExcelApp.ActiveWorkbook
1153            CType(Wb.Sheets(index), Excel.Worksheet).Delete()
1154
1155            CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1156        End Sub
1157
1158        '''
1159        ' This method adds a worksheet named "___CAFE_AUTO_GENERATED_SHEET___" as the first sheet on the
1160        ' active workbook, and removes all other worksheets.
1161        '
```

```
1162      Public Sub RemoveAllWorksheets(Optional ByRef AddNewSheet As Boolean = False)
1163          CAFE.Profiling.FunctionStart()
1164
1165          Dim Wb As Excel.Workbook = ExcelApp.ActiveWorkbook
1166          Dim i As Integer
1167
1168          If AddNewSheet Then
1169              AddWorkSheet(1, "___CAFE_AUTO_GENERATED_SHEET___")
1170          Else
1171              NameWorkSheet(1, "___CAFE_AUTO_GENERATED_SHEET___")
1172          End If
1173
1174          For i = Wb.Sheets.Count To 2 Step -1
1175              RemoveWorksheet(i)
1176          Next i
1177
1178          CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1179      End Sub
1180
1181      '''
1182      ' This method sets the name of the worksheet at the specified index.  If the worksheet ↙
          name
1183      ' cannot be set, an error message will be displayed.
1184      '
1185      ' Params:
1186      '   - index         - the index of the worksheet to which to assign the given name
1187      '   - name          - the name to assign to the worksheet at the given index
1188      '
1189      Public Sub NameWorkSheet(ByRef index As Integer, ByRef name As String)
1190          CAFE.Profiling.FunctionStart()
1191
1192          Dim Wb As Excel.Workbook = ExcelApp.ActiveWorkbook
1193          Dim Ws As Excel.Sheets = Wb.Worksheets
1194
1195          name = name.Trim()
1196
1197          If name = "" Then
1198              If DBUG Then
1199                  MsgBox("In NameWorkSheet(index, name): " & _
1200                         "Cannot rename worksheet.  Name cannot be empty.")
1201              End If
1202          Else
1203              Try
1204                  CType(Ws(index), Excel.Worksheet).Name = name
1205              Catch ex As Exception
1206                  If DBUG Then
1207                      MsgBox("In NameWorkSheet(index, name): " & ex.Message)
1208                  End If
1209
1210                  ' try adding worksheet again, with the appended name
1211                  name = name & "_" & Now.Hour & "." & Now.Minute
1212                  NameWorkSheet(index, name)
1213              End Try
1214          End If
1215
1216          CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1217      End Sub
1218
1219      '''
1220      ' This method copies a worksheet with the given name from the source workbook to the  ↙
          destination
1221      ' workbook.  The worksheet is inserted at the beginning.
1222      '
1223      ' Params:
1224      '   src             - the source workbook, from which to copy the worksheet
1225      '   dst             - the destination workbook, where the worksheet should be copied  ↙
          to
1226      '   name            - the name of the worksheet to copy
1227      '
1228      Public Sub CopyWorksheet(ByRef src As Excel.Workbook, _
```

```
1229                                ByRef dst As Excel.Workbook, _
1230                                ByRef name As String)
1231            '
1232            CAFE.Profiling.FunctionStart()
1233
1234            '
1235            CopyWorksheet(src, dst, 1, name)
1236
1237            CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1238        End Sub
1239
1240        '''
1241        ' This method copies a worksheet with the given name from the source workbook to the destination
1242        ' workbook.  The worksheet is inserted at the given index.
1243        '
1244        ' Params:
1245        '   src             - the source workbook, from which to copy the worksheet
1246        '   dst             - the destination workbook, where the worksheet should be copied to
1247        '   index           - the index at which to insert the copied worksheet
1248        '   name            - the name of the worksheet to copy
1249        '
1250        Public Sub CopyWorksheet(ByRef src As Excel.Workbook, _
1251                                 ByRef dst As Excel.Workbook, _
1252                                 ByRef index As Integer, _
1253                                 ByRef name As String)
1254            '
1255            CAFE.Profiling.FunctionStart()
1256
1257            '
1258            'src.Activate()
1259            'CType(src.Sheets(name), Excel.Worksheet).Select()
1260
1261            If index <= 1 Then
1262                ' before first sheet
1263                CType(src.Sheets(name), Excel.Worksheet).Copy(Before:=dst.Sheets(1))
1264            ElseIf index > dst.Worksheets.Count Then
1265                ' at the end of last sheet
1266                CType(src.Sheets(name), Excel.Worksheet).Copy(After:=dst.Sheets(dst.Sheets.Count))
1267            Else
1268                ' before the sheet at given index
1269                CType(src.Sheets(name), Excel.Worksheet).Copy(Before:=dst.Sheets(index))
1270            End If
1271
1272            'dst.Activate()
1273
1274            CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1275        End Sub
1276
1277    #End Region
1278
1279
1280    #Region " Methods for drawing graphs and charts "
1281
1282        '''
1283        ' This method provides limited support for drawing excel charts.  This method draws a single
1284        ' chart with one series range.
1285        '
1286        ' Params:
1287        '   chartType       - the excel chart type to draw; supported charts include, but not limited to,
1288        '                     line, stacked line, bar, and stacked bar graphs; pie charts are not
1289        '                     supported
1290        '   sheetName       - the name of the excel worksheet for which to draw the graph
1291        '   chartName       - the name of the resultant chart
1292        '   categoryRange   - the data range for the category axis, e.g. the x-axis, of the
```

```
    excel chart
1293    '   seriesRange     - a series range from which the chart will be produced
1294    '   seriesNames     - the name to assign to the series range
1295    '   chartTitle      - optional; the chart's title to display at the top of the page
1296    '   categoryTitle   - optional; the title of the chart's category axis, which is displayed below
1297    '                     the chart's x-axes values
1298    '   valueTitle      - optional; the title of the chart's value axis, which is displayed to the
1299    '                     left of the chart's y-axes values
1300    '
1301    Public Sub DrawChart(ByRef chartType As Excel.XlChartType, _
1302                         ByRef sheetName As String, _
1303                         ByRef chartName As String, _
1304                         ByRef categoryRange As Excel.Range, _
1305                         ByRef seriesRange As Excel.Range, _
1306                         ByRef seriesNames As String, _
1307                         Optional ByRef chartTitle As String = "", _
1308                         Optional ByRef categoryTitle As String = "", _
1309                         Optional ByRef valueTitle As String = "")
1310        '
1311        CAFE.Profiling.FunctionStart()
1312
1313        '
1314        ' draws a single chart
1315        Dim seriesRangeArr() As Excel.Range = {seriesRange}
1316        Dim seriesNamesArr() As String = {seriesNames}
1317        ' call overloaded DrawChart() method
1318        DrawChart(chartType, sheetName, chartName, 1, categoryRange, seriesRangeArr, seriesNamesArr, chartTitle, categoryTitle, valueTitle)
1319
1320        CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1321    End Sub
1322
1323    '''
1324    ' This method provides limited support for drawing excel charts.  This method draws a single
1325    ' chart with the maximum number of range series determined by the nSeries parameter.
1326    '
1327    ' Params:
1328    '   chartType       - the excel chart type to draw; supported charts include, but not limited to,
1329    '                     line, stacked line, bar, and stacked bar graphs; pie charts are not
1330    '                     supported
1331    '   sheetName       - the name of the excel worksheet for which to draw the graph
1332    '   chartName       - the name of the resultant chart
1333    '   nSeries         - the total number of series that will be displayed on the resultant chart
1334    '   categoryRange   - the data range for the category axis, e.g. the x-axis, of the excel chart
1335    '   seriesRange     - an array of series ranges from which the chart will be produced; all ranges
1336    '     [array]         in the seriesRange array will be displayed on a single chart; the length
1337    '                     of the seriesRange array must be equal to nSeries
1338    '   seriesNames     - an array of series names; the name to assign to each individual range; the
1339    '     [array]         length of seriesNames array must be equal to nSeries
1340    '   chartTitle      - optional; the chart's title to display at the top of the page
1341    '   categoryTitle   - optional; the title of the chart's category axis, which is displayed below
1342    '                     the chart's x-axes values
1343    '   valueTitle      - optional; the title of the chart's value axis, which is displayed to the
1344    '                     left of the chart's y-axes values
1345    '
1346    Public Sub DrawChart(ByRef chartType As Excel.XlChartType, _
1347                         ByRef sheetName As String, _
1348                         ByRef chartName As String, _
```

```
1349                            ByRef nSeries As Integer, _
1350                            ByRef categoryRange As Excel.Range, _
1351                            ByRef seriesRange() As Excel.Range, _
1352                            ByRef seriesNames() As String, _
1353                            Optional ByRef chartTitle As String = "", _
1354                            Optional ByRef categoryTitle As String = "", _
1355                            Optional ByRef valueTitle As String = "")
1356        '
1357        CAFE.Profiling.FunctionStart()
1358
1359        '
1360        ' draws a collection of charts
1361        Dim i As Integer
1362        Dim wb As Excel.Workbook
1363        Dim sheet As Excel.Worksheet
1364        Dim chart As Excel.Chart
1365
1366        Dim seriesX1 As Integer
1367        Dim seriesY1 As Integer
1368        Dim seriesX2 As Integer
1369        Dim seriesY2 As Integer
1370
1371        Dim categoryX1 As Integer = categoryRange.Row
1372        Dim categoryY1 As Integer = categoryRange.Column
1373        Dim categoryX2 As Integer = categoryRange.Rows.Count + categoryX1 - 1
1374        Dim categoryY2 As Integer = categoryRange.Columns.Count + categoryY1 - 1
1375
1376        wb = ExcelApp.ActiveWorkbook
1377        sheet = wb.Sheets(sheetName)
1378
1379        wb.Charts.Add()
1380        chart = wb.ActiveChart
1381        chart.ChartType = chartType
1382
1383        ' set source data and delete default series
1384        chart.SetSourceData(sheet.Range("A1"), Excel.XlRowCol.xlColumns)
1385        chart.SeriesCollection(1).Delete()
1386
1387        ' add new series and set its values
1388        For i = 1 To nSeries
1389            ' get series coordinates from seriesRange(i)
1390            seriesX1 = seriesRange(i - 1).Row
1391            seriesY1 = seriesRange(i - 1).Column
1392            seriesX2 = seriesRange(i - 1).Rows.Count + seriesX1 - 1
1393            seriesY2 = seriesRange(i - 1).Columns.Count + seriesY1 - 1
1394
1395            ' add new series
1396            chart.SeriesCollection.NewSeries()
1397            ' set the series' name
1398            chart.SeriesCollection(i).Name() = "=""" & seriesNames(i - 1) & """"
1399            ' set values
1400            chart.SeriesCollection(i).Values = "=" & sheetName &
1401                                               "!R" & seriesX1 & "C" & seriesY1 & _
1402                                               ":R" & seriesX2 & "C" & seriesY2
1403        Next i
1404
1405        ' set x values
1406        chart.SeriesCollection(1).XValues = "=" & sheetName & _
1407                                            "!R" & categoryX1 & "C" & categoryY1 & _
1408                                            ":R" & categoryX2 & "C" & categoryY2
1409
1410        chart.Location(Excel.XlChartLocation.xlLocationAsNewSheet, chartName)
1411        With chart
1412            If chartTitle = "" Then
1413                .HasTitle = False
1414            Else
1415                .HasTitle = True
1416                .ChartTitle.Characters.Text = chartTitle
1417            End If
1418
```

```
1419                If categoryTitle = "" Then
1420                    .Axes(Excel.XlAxisType.xlCategory, Excel.XlAxisGroup.xlPrimary).HasTitle = ↙
     False
1421                Else
1422                    .Axes(Excel.XlAxisType.xlCategory, Excel.XlAxisGroup.xlPrimary).HasTitle = ↙
     True
1423                    .Axes(Excel.XlAxisType.xlCategory, Excel.XlAxisGroup.xlPrimary).AxisTitle. ↙
    Characters.Text = categoryTitle
1424                End If
1425
1426                If valueTitle = "" Then
1427                    .Axes(Excel.XlAxisType.xlValue, Excel.XlAxisGroup.xlPrimary).HasTitle =    ↙
    False
1428                Else
1429                    .Axes(Excel.XlAxisType.xlValue, Excel.XlAxisGroup.xlPrimary).HasTitle =    ↙
    True
1430                    .Axes(Excel.XlAxisType.xlValue, Excel.XlAxisGroup.xlPrimary).AxisTitle.    ↙
    Characters.Text = valueTitle
1431                End If
1432            End With
1433
1434            CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1435        End Sub
1436
1437    #End Region
1438
1439
1440    #Region " Helper Methods "
1441
1442        '''
1443        ' This method converts the given colorObj object to the equivalent System.Drawing.     ↙
        Color instance.
1444        ' If the colorObj cannot be converted, then return value is System.Drawing.Color.Black ↙
        .
1445        '
1446        ' Params:
1447        '   colorObj      - the color object from which to determine the System.Drawing.     ↙
        Color instance
1448        '
1449        ' Returns:
1450        '   - the System.Drawing.Color instance parsed from the colorObj.object, or
1451        '     System.Drawing.Color.Black, if colorObj cannot be parsed
1452        '
1453        Private Function GetColor(ByRef colorObj As Object) As System.Drawing.Color
1454            'CAFE.Profiling.FunctionStart()
1455
1456            Dim color As System.Drawing.Color
1457
1458            If colorObj.GetType.Equals(GetType(String)) Then
1459                Try
1460                    color = color.FromName(colorObj.ToString)
1461                Catch ex As Exception
1462                    If DBUG Then
1463                        MsgBox("In GetColor(colorObj): " & ex.Message)
1464                    End If
1465
1466                    color = System.Drawing.Color.Black
1467                End Try
1468            Else
1469                Try
1470                    color = CType(colorObj, System.Drawing.Color)
1471                Catch ex As Exception
1472                    If DBUG Then
1473                        MsgBox("In GetColor(colorObj): " & ex.Message)
1474                    End If
1475
1476                    color = System.Drawing.Color.Black
1477                End Try
1478            End If
1479
```

```
1480            'CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1481            Return color
1482        End Function
1483
1484        '''
1485        ' This method converts the specified cellIndex to an excel cell object.  If the specified cellIndex
1486        ' is numeric, it is converted to a corresponding excel "letter" column index.
1487        '
1488        ' Params:
1489        '   cellIndex       - the cell index to convert to an excel "letter" column index
1490        '
1491        ' Returns:
1492        '   - if an integer, the converted cellIndex, otherwise the original cellIndex parameter
1493        '
1494        Private Function ConvertExcelColumnIndex(ByVal cellIndex As Object) As Object
1495            'CAFE.Profiling.FunctionStart()
1496
1497            Dim idx As Integer
1498
1499            If IsNumeric(cellIndex) Then
1500                idx = Integer.Parse(cellIndex)
1501
1502                If idx > 26 Then
1503                    cellIndex = Chr((64 + (idx \ 26))) + Chr(64 + (idx Mod 26))
1504                Else
1505                    cellIndex = Chr(64 + idx)
1506                End If
1507            End If
1508
1509            'CAFE.Profiling.FunctionEnd(System.Reflection.MethodBase.GetCurrentMethod())
1510            Return cellIndex
1511        End Function
1512
1513    #End Region
1514
1515    End Module
1516
```

```
1596        ' calculate emission rates for gas = (Conv% * ConvRate) + ((1-Conv%) * RFGRate)
1597        Dim emisRateNOXLDGV(MY, CY) As Double
1598        Dim emisRateNOXLDGT1(MY, CY) As Double
1599        Dim emisRateNOXLDGT2(MY, CY) As Double
1600        Dim emisRateNOXHDGV2b(MY, CY) As Double
1601
1602        For i = 0 To CY
1603            For j = 0 To MY
1604                If i >= j Then
1605                    emisRateNOXLDGV(j, i) = (CAFE.Parameters.noxGasLDGV(j, Math.Min(i, 25)) ↙
    * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters. ↙
    noxGasRFGLDGV(j, Math.Min(i, 25)))
1606                    emisRateNOXLDGT1(j, i) = (CAFE.Parameters.noxGasLDGT1(j, Math.Min(i, ↙
    25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters ↙
    .noxGasRFGLDGT1(j, Math.Min(i, 25)))
1607                    emisRateNOXLDGT2(j, i) = (CAFE.Parameters.noxGasLDGT2(j, Math.Min(i, ↙
    25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters ↙
    .noxGasRFGLDGT2(j, Math.Min(i, 25)))
1608                    emisRateNOXHDGV2b(j, i) = (CAFE.Parameters.noxGasHDGV2b(j, Math.Min(i, ↙
    25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters ↙
    .noxGasRFGHDGV2b(j, Math.Min(i, 25)))
1609                End If
1610            Next j
1611        Next i
1612
1613        Dim emisOpNOXLDGV(MY, CY) As Double
1614        Dim emisOpNOXLDGT1(MY, CY) As Double
1615        Dim emisOpNOXLDGT2(MY, CY) As Double
1616        Dim emisOpNOXHDGV2b(MY, CY) As Double
1617
1618        Dim emisOpNOXLDDV(MY, CY) As Double
1619        Dim emisOpNOXLDDV1(MY, CY) As Double
1620        Dim emisOpNOXLDDV2(MY, CY) As Double
1621        Dim emisOpNOXHDDV2b(MY, CY) As Double
1622
1623        For i = 0 To CY
1624            For j = 0 To MY
1625                If i >= j Then
1626                    emisOpNOXLDGV(j, i) = (VMTLDGV(j, i) * emisRateNOXLDGV(j, i)) * (10 ^ 6) ↙
     / (454 * 2000 * 1000)
1627                    emisOpNOXLDGT1(j, i) = (VMTLDGT1(j, i) * emisRateNOXLDGT1(j, i)) * (10 ^ ↙
     6) / (454 * 2000 * 1000)
1628                    emisOpNOXLDGT2(j, i) = (VMTLDGT2(j, i) * emisRateNOXLDGT2(j, i)) * (10 ^ ↙
     6) / (454 * 2000 * 1000)
1629                    emisOpNOXHDGV2b(j, i) = (VMTHDGV2b(j, i) * emisRateNOXHDGV2b(j, i)) * ↙
    (10 ^ 6) / (454 * 2000 * 1000)
1630
1631                    emisOpNOXLDDV(j, i) = (VMTLDDV(j, i) * CAFE.Parameters.noxDiesLDDV(j, ↙
    Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1632                    emisOpNOXLDDV1(j, i) = (VMTLDDV1(j, i) * CAFE.Parameters.noxDiesLDDT1(j, ↙
     Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1633                    emisOpNOXLDDV2(j, i) = (VMTLDDV2(j, i) * CAFE.Parameters.noxDiesLDDT2(j, ↙
     Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1634                    emisOpNOXHDDV2b(j, i) = (VMTHDDV2b(j, i) * CAFE.Parameters.noxDiesHDDV2b ↙
    (j, Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1635                End If
1636            Next
1637        Next
1638
1639        'Total ("Well to Wheel") Nitrous Oxide (NOX) Emissions
1640        Dim emisTotNOXLDGV(MY, CY) As Double
1641        Dim emisTotNOXLDGT1(MY, CY) As Double
1642        Dim emisTotNOXLDGT2(MY, CY) As Double
1643        Dim emisTotNOXHDGV2b(MY, CY) As Double
1644
1645        Dim emisTotNOXLDDV(MY, CY) As Double
1646        Dim emisTotNOXLDDV1(MY, CY) As Double
1647        Dim emisTotNOXLDDV2(MY, CY) As Double
1648        Dim emisTotNOXHDDV2b(MY, CY) As Double
```

```
1649
1650        'TOTALS:
1651        Dim gasEmisTotNOXTotal(MY, CY) As Double
1652        Dim diesEmisTotNOXTotal(MY, CY) As Double
1653
1654        For i = 0 To CY
1655            For j = 0 To MY
1656                If i >= j Then
1657                    emisTotNOXLDGV(j, i) = emisExtrNOXLDGV(j, i) + emisTransNOXLDGV(j, i) + emisRefNOXLDGV(j, i) + emisTSDNOXLDGV(j, i) + emisOpNOXLDGV(j, i)
1658                    emisTotNOXLDGT1(j, i) = emisExtrNOXLDGT1(j, i) + emisTransNOXLDGT1(j, i) + emisRefNOXLDGT1(j, i) + emisTSDNOXLDGT1(j, i) + emisOpNOXLDGT1(j, i)
1659                    emisTotNOXLDGT2(j, i) = emisExtrNOXLDGT2(j, i) + emisTransNOXLDGT2(j, i) + emisRefNOXLDGT2(j, i) + emisTSDNOXLDGT2(j, i) + emisOpNOXLDGT2(j, i)
1660                    emisTotNOXHDGV2b(j, i) = emisExtrNOXHDGV2b(j, i) + emisTransNOXHDGV2b(j, i) + emisRefNOXHDGV2b(j, i) + emisTSDNOXHDGV2b(j, i) + emisOpNOXHDGV2b(j, i)
1661
1662                    emisTotNOXLDDV(j, i) = emisExtrNOXLDDV(j, i) + emisTransNOXLDDV(j, i) + emisRefNOXLDDV(j, i) + emisTSDNOXLDDV(j, i) + emisOpNOXLDDV(j, i)
1663                    emisTotNOXLDDV1(j, i) = emisExtrNOXLDDV1(j, i) + emisTransNOXLDDV1(j, i) + emisRefNOXLDDV1(j, i) + emisTSDNOXLDDV1(j, i) + emisOpNOXLDDV1(j, i)
1664                    emisTotNOXLDDV2(j, i) = emisExtrNOXLDDV2(j, i) + emisTransNOXLDDV2(j, i) + emisRefNOXLDDV2(j, i) + emisTSDNOXLDDV2(j, i) + emisOpNOXLDDV2(j, i)
1665                    emisTotNOXHDDV2b(j, i) = emisExtrNOXHDDV2b(j, i) + emisTransNOXHDDV2b(j, i) + emisRefNOXHDDV2b(j, i) + emisTSDNOXHDDV2b(j, i) + emisOpNOXHDDV2b(j, i)
1666
1667                    'TOTALS:
1668                    gasEmisTotNOXTotal(j, i) = emisTotNOXLDGV(j, i) + emisTotNOXLDGT1(j, i) + emisTotNOXLDGT2(j, i) + emisTotNOXHDGV2b(j, i)
1669                    diesEmisTotNOXTotal(j, i) = emisTotNOXLDDV(j, i) + emisTotNOXLDDV1(j, i) + emisTotNOXLDDV2(j, i) + emisTotNOXHDDV2b(j, i)
1670                End If
1671            Next
1672        Next
1673
1674        'Economic Benefit/Cost of Total Nitrous Oxide (NOX) Emissions
1675        Dim damageEmisTotNOXLDGV(MY, CY) As Double
1676        Dim damageEmisTotNOXLDGT1(MY, CY) As Double
1677        Dim damageEmisTotNOXLDGT2(MY, CY) As Double
1678        Dim damageEmisTotNOXHDGV2b(MY, CY) As Double
1679
1680        Dim damageEmisTotNOXLDDV(MY, CY) As Double
1681        Dim damageEmisTotNOXLDDV1(MY, CY) As Double
1682        Dim damageEmisTotNOXLDDV2(MY, CY) As Double
1683        Dim damageEmisTotNOXHDDV2b(MY, CY) As Double
1684
1685        Dim discDamageEmisTotNOXLDGV(MY, CY) As Double
1686        Dim discDamageEmisTotNOXLDGT1(MY, CY) As Double
1687        Dim discDamageEmisTotNOXLDGT2(MY, CY) As Double
1688        Dim discDamageEmisTotNOXHDGV2b(MY, CY) As Double
1689
1690        Dim discDamageEmisTotNOXLDDV(MY, CY) As Double
1691        Dim discDamageEmisTotNOXLDDV1(MY, CY) As Double
1692        Dim discDamageEmisTotNOXLDDV2(MY, CY) As Double
1693        Dim discDamageEmisTotNOXHDDV2b(MY, CY) As Double
1694
1695        'TOTALS:
1696        Dim gasDamageEmisTotNOXTotal(MY, CY) As Double
1697        Dim diesDamageEmisTotNOXTotal(MY, CY) As Double
1698
1699        Dim discGasDamageEmisTotNOXTotal(MY, CY) As Double
1700        Dim discDiesDamageEmisTotNOXTotal(MY, CY) As Double
1701
1702        For i = 0 To CY
1703            For j = 0 To MY
1704                If i >= j Then
1705                    damageEmisTotNOXLDGV(j, i) = emisTotNOXLDGV(j, i) * CAFE.Parameters.damageCostNOX / 1000
1706                    damageEmisTotNOXLDGT1(j, i) = emisTotNOXLDGT1(j, i) * CAFE.Parameters.
```

```
damageCostNOX / 1000
                        damageEmisTotNOXLDGT2(j, i) = emisTotNOXLDGT2(j, i) * CAFE.Parameters.damageCostNOX / 1000
                        damageEmisTotNOXHDGV2b(j, i) = emisTotNOXHDGV2b(j, i) * CAFE.Parameters.damageCostNOX / 1000

                        damageEmisTotNOXLDDV(j, i) = emisTotNOXLDDV(j, i) * CAFE.Parameters.damageCostNOX / 1000
                        damageEmisTotNOXLDDV1(j, i) = emisTotNOXLDDV1(j, i) * CAFE.Parameters.damageCostNOX / 1000
                        damageEmisTotNOXLDDV2(j, i) = emisTotNOXLDDV2(j, i) * CAFE.Parameters.damageCostNOX / 1000
                        damageEmisTotNOXHDDV2b(j, i) = emisTotNOXHDDV2b(j, i) * CAFE.Parameters.damageCostNOX / 1000

                        discDamageEmisTotNOXLDGV(j, i) = damageEmisTotNOXLDGV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotNOXLDGT1(j, i) = damageEmisTotNOXLDGT1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotNOXLDGT2(j, i) = damageEmisTotNOXLDGT2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotNOXHDGV2b(j, i) = damageEmisTotNOXHDGV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                        discDamageEmisTotNOXLDDV(j, i) = damageEmisTotNOXLDDV(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotNOXLDDV1(j, i) = damageEmisTotNOXLDDV1(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotNOXLDDV2(j, i) = damageEmisTotNOXLDDV2(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)
                        discDamageEmisTotNOXHDDV2b(j, i) = damageEmisTotNOXHDDV2b(j, i) * ((1 + CAFE.Parameters.discountRate) ^ -i)

                        'TOTALS:
                        gasDamageEmisTotNOXTotal(j, i) = damageEmisTotNOXLDGV(j, i) + damageEmisTotNOXLDGT1(j, i) + damageEmisTotNOXLDGT2(j, i) + damageEmisTotNOXHDGV2b(j, i)
                        diesDamageEmisTotNOXTotal(j, i) = damageEmisTotNOXLDDV(j, i) + damageEmisTotNOXLDDV1(j, i) + damageEmisTotNOXLDDV2(j, i) + damageEmisTotNOXHDDV2b(j, i)

                        discGasDamageEmisTotNOXTotal(j, i) = discDamageEmisTotNOXLDGV(j, i) + discDamageEmisTotNOXLDGT1(j, i) + discDamageEmisTotNOXLDGT2(j, i) + discDamageEmisTotNOXHDGV2b(j, i)
                        discDiesDamageEmisTotNOXTotal(j, i) = discDamageEmisTotNOXLDDV(j, i) + discDamageEmisTotNOXLDDV1(j, i) + discDamageEmisTotNOXLDDV2(j, i) + discDamageEmisTotNOXHDDV2b(j, i)

                    End If
                Next
            Next



            ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
            'PARTICULATE MATTER (PM)
            'average PM upstream emissions (grams/MBTU) vars:
            Dim avgUpEmisExtrPM As Double
            Dim avgUpEmisTransPM As Double
            Dim avgUpEmisRefPM As Double
            Dim avgUpEmisTSDPM As Double

            Dim avgUpEmisExtrDiesPM As Double
            Dim avgUpEmisTransDiesPM As Double
            Dim avgUpEmisRefDiesPM As Double
            Dim avgUpEmisTSDDiesPM As Double

            'calculate PM average upstream emissions (grams/MBTU)
            avgUpEmisExtrPM = (CAFE.Parameters.totExtrConvGasPM * CAFE.Parameters.convGasMix) + (CAFE.Parameters.totExtrFedGasPM * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.
```

```
    totExtrCaliGasPM * CAFE.Parameters.caliGasMix)
1753        avgUpEmisTransPM = (CAFE.Parameters.totTransConvGasPM * CAFE.Parameters.convGasMix)
    + (CAFE.Parameters.totTransFedGasPM * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.
    totTransCaliGasPM * CAFE.Parameters.caliGasMix)
1754        avgUpEmisRefPM = (CAFE.Parameters.totRefConvGasPM * CAFE.Parameters.convGasMix) +
    (CAFE.Parameters.totRefFedGasPM * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.
    totRefCaliGasPM * CAFE.Parameters.caliGasMix)
1755        avgUpEmisTSDPM = (CAFE.Parameters.totTSDConvGasPM * CAFE.Parameters.convGasMix) +
    (CAFE.Parameters.totTSDFedGasPM * CAFE.Parameters.fedGasMix) + (CAFE.Parameters.
    totTSDCaliGasPM * CAFE.Parameters.caliGasMix)
1756
1757        avgUpEmisExtrDiesPM = (CAFE.Parameters.totExtrConvDiesPM * CAFE.Parameters.
    convDiesMix) + (CAFE.Parameters.totExtrLowDiesPM * CAFE.Parameters.lowDiesMix)
1758        avgUpEmisTransDiesPM = (CAFE.Parameters.totTransConvDiesPM * CAFE.Parameters.
    convDiesMix) + (CAFE.Parameters.totTransLowDiesPM * CAFE.Parameters.lowDiesMix)
1759        avgUpEmisRefDiesPM = (CAFE.Parameters.totRefConvDiesPM * CAFE.Parameters.
    convDiesMix) + (CAFE.Parameters.totRefLowDiesPM * CAFE.Parameters.lowDiesMix)
1760        avgUpEmisTSDDiesPM = (CAFE.Parameters.totTSDConvDiesPM * CAFE.Parameters.
    convDiesMix) + (CAFE.Parameters.totTSDLowDiesPM * CAFE.Parameters.lowDiesMix)
1761
1762        'adjusted PM average upstream emissions (grams/gal) vars:
1763        Dim adjUpEmisExtrPM As Double
1764        Dim adjUpEmisTransPM As Double
1765        Dim adjUpEmisRefPM As Double
1766        Dim adjUpEmisTSDPM As Double
1767
1768        Dim adjUpEmisExtrDiesPM As Double
1769        Dim adjUpEmisTransDiesPM As Double
1770        Dim adjUpEmisRefDiesPM As Double
1771        Dim adjUpEmisTSDDiesPM As Double
1772
1773        'calculate PM adjusted average upstream emissions (grams/gal):
1774        adjUpEmisExtrPM = avgUpEmisExtrPM * (avgGasEng / 10 ^ 6)
1775        adjUpEmisTransPM = avgUpEmisTransPM * (avgGasEng / 10 ^ 6)
1776        adjUpEmisRefPM = avgUpEmisRefPM * (avgGasEng / 10 ^ 6)
1777        adjUpEmisTSDPM = avgUpEmisTSDPM * (avgGasEng / 10 ^ 6)
1778
1779        adjUpEmisExtrDiesPM = avgUpEmisExtrDiesPM * (avgDiesEng / 10 ^ 6)
1780        adjUpEmisTransDiesPM = avgUpEmisTransDiesPM * (avgDiesEng / 10 ^ 6)
1781        adjUpEmisRefDiesPM = avgUpEmisRefDiesPM * (avgDiesEng / 10 ^ 6)
1782        adjUpEmisTSDDiesPM = avgUpEmisTSDDiesPM * (avgDiesEng / 10 ^ 6)
1783
1784        'Particulate Matter (PM) Emissions from Domestic Crude Petroleum Extraction
1785        Dim emisExtrPMLDGV(MY, CY) As Double
1786        Dim emisExtrPMLDGT1(MY, CY) As Double
1787        Dim emisExtrPMLDGT2(MY, CY) As Double
1788        Dim emisExtrPMHDGV2b(MY, CY) As Double
1789
1790        Dim emisExtrPMLDDV(MY, CY) As Double
1791        Dim emisExtrPMLDDV1(MY, CY) As Double
1792        Dim emisExtrPMLDDV2(MY, CY) As Double
1793        Dim emisExtrPMHDDV2b(MY, CY) As Double
1794
1795        For i = 0 To CY
1796            For j = 0 To MY
1797                If i >= j Then
1798                    emisExtrPMLDGV(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
    gasLowDomRefDomCrude) * (adjUpEmisExtrPM * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1799                    emisExtrPMLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
    gasLowDomRefDomCrude) * (adjUpEmisExtrPM * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 *
    1000)
1800                    emisExtrPMLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters.
    gasLowDomRefDomCrude) * (adjUpEmisExtrPM * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 *
    1000)
1801                    emisExtrPMHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef * CAFE.Parameters
    .gasLowDomRefDomCrude) * (adjUpEmisExtrPM * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 *
    1000)
1802
1803                    emisExtrPMLDDV(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters.
```

```
diesLowDomRefDomCrude) * (adjUpEmisExtrDiesPM * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisExtrPMLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesPM * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisExtrPMLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesPM * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisExtrPMHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef * CAFE.Parameters.diesLowDomRefDomCrude) * (adjUpEmisExtrDiesPM * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                End If
            Next
        Next

        'Particulate Matter (PM) Emissions from Domestic Crude Petroleum Transportation
        Dim emisTransPMLDGV(MY, CY) As Double
        Dim emisTransPMLDGT1(MY, CY) As Double
        Dim emisTransPMLDGT2(MY, CY) As Double
        Dim emisTransPMHDGV2b(MY, CY) As Double

        Dim emisTransPMLDDV(MY, CY) As Double
        Dim emisTransPMLDDV1(MY, CY) As Double
        Dim emisTransPMLDDV2(MY, CY) As Double
        Dim emisTransPMHDDV2b(MY, CY) As Double

        For i = 0 To CY
            For j = 0 To MY
                If i >= j Then
                    emisTransPMLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisTransPM * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisTransPMLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisTransPM * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisTransPMLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisTransPM * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisTransPMHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisTransPM * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)

                    emisTransPMLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) * (adjUpEmisTransDiesPM * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisTransPMLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) * (adjUpEmisTransDiesPM * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisTransPMLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) * (adjUpEmisTransDiesPM * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisTransPMHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) * (adjUpEmisTransDiesPM * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                End If
            Next
        Next

        ' Particulate Matter (PM) Emissions from Domestic Crude Petroleum Refining to Produce Gasoline
        Dim emisRefPMLDGV(MY, CY) As Double
        Dim emisRefPMLDGT1(MY, CY) As Double
        Dim emisRefPMLDGT2(MY, CY) As Double
        Dim emisRefPMHDGV2b(MY, CY) As Double

        Dim emisRefPMLDDV(MY, CY) As Double
        Dim emisRefPMLDDV1(MY, CY) As Double
        Dim emisRefPMLDDV2(MY, CY) As Double
        Dim emisRefPMHDDV2b(MY, CY) As Double

        For i = 0 To CY
            For j = 0 To MY
                If i >= j Then
                    emisRefPMLDGV(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefPM * galLDGV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
                    emisRefPMLDGT1(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefPM * galLDGT1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
```

```
1854                    emisRefPMLDGT2(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefPM
    * galLDGT2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1855                    emisRefPMHDGV2b(j, i) = (CAFE.Parameters.gasLowDomRef) * (adjUpEmisRefPM
     * galHDGV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1856
1857                    emisRefPMLDDV(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesPM * galLDDV(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1858                    emisRefPMLDDV1(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesPM * galLDDV1(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1859                    emisRefPMLDDV2(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesPM * galLDDV2(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1860                    emisRefPMHDDV2b(j, i) = (CAFE.Parameters.diesLowDomRef) *
    (adjUpEmisRefDiesPM * galHDDV2b(j, i) * 10 ^ 6) / (454 * 2000 * 1000)
1861                End If
1862            Next
1863        Next
1864
1865        'Particulate Matter (PM) Emissions from Gasoline Transportation, Storage, and
    Distribution
1866        Dim emisTSDPMLDGV(MY, CY) As Double
1867        Dim emisTSDPMLDGT1(MY, CY) As Double
1868        Dim emisTSDPMLDGT2(MY, CY) As Double
1869        Dim emisTSDPMHDGV2b(MY, CY) As Double
1870
1871        Dim emisTSDPMLDDV(MY, CY) As Double
1872        Dim emisTSDPMLDDV1(MY, CY) As Double
1873        Dim emisTSDPMLDDV2(MY, CY) As Double
1874        Dim emisTSDPMHDDV2b(MY, CY) As Double
1875
1876        For i = 0 To CY
1877            For j = 0 To MY
1878                If i >= j Then
1879                    emisTSDPMLDGV(j, i) = (adjUpEmisTSDPM * galLDGV(j, i) * 10 ^ 6) / (454 *
     2000 * 1000)
1880                    emisTSDPMLDGT1(j, i) = (adjUpEmisTSDPM * galLDGT1(j, i) * 10 ^ 6) / (454
    * 2000 * 1000)
1881                    emisTSDPMLDGT2(j, i) = (adjUpEmisTSDPM * galLDGT2(j, i) * 10 ^ 6) / (454
     * 2000 * 1000)
1882                    emisTSDPMHDGV2b(j, i) = (adjUpEmisTSDPM * galHDGV2b(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1883
1884                    emisTSDPMLDDV(j, i) = (adjUpEmisTSDDiesPM * galLDDV(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1885                    emisTSDPMLDDV1(j, i) = (adjUpEmisTSDDiesPM * galLDDV1(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1886                    emisTSDPMLDDV2(j, i) = (adjUpEmisTSDDiesPM * galLDDV2(j, i) * 10 ^ 6) /
    (454 * 2000 * 1000)
1887                    emisTSDPMHDDV2b(j, i) = (adjUpEmisTSDDiesPM * galHDDV2b(j, i) * 10 ^ 6)
    / (454 * 2000 * 1000)
1888                End If
1889            Next
1890        Next
1891
1892
1893        ''Particulate Matter (PM) Emissions from Vehicle Operation ("Tank to Wheel")
1894        ' calculate emission rates for gas = (Conv% * ConvRate) + ((1-Conv%) * RFGRate)
1895        Dim emisRatePMLDGV(MY, CY) As Double
1896        Dim emisRatePMLDGT1(MY, CY) As Double
1897        Dim emisRatePMLDGT2(MY, CY) As Double
1898        Dim emisRatePMHDGV2b(MY, CY) As Double
1899
1900        For i = 0 To CY
1901            For j = 0 To MY
1902                If i >= j Then
1903                    emisRatePMLDGV(j, i) = (CAFE.Parameters.pmGasLDGV(j, Math.Min(i, 25)) *
    CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters.
    pmGasRFGLDGV(j, Math.Min(i, 25)))
1904                    emisRatePMLDGT1(j, i) = (CAFE.Parameters.pmGasLDGT1(j, Math.Min(i, 25))
    * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters.
```

```
pmGasRFGLDGT1(j, Math.Min(i, 25)))
1905                    emisRatePMLDGT2(j, i) = (CAFE.Parameters.pmGasLDGT2(j, Math.Min(i, 25)) ↙
* CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters. ↙
pmGasRFGLDGT2(j, Math.Min(i, 25)))
1906                    emisRatePMHDGV2b(j, i) = (CAFE.Parameters.pmGasHDGV2b(j, Math.Min(i, ↙
25)) * CAFE.Parameters.convGasMix) + ((1 - CAFE.Parameters.convGasMix) * CAFE.Parameters ↙
.pmGasRFGHDGV2b(j, Math.Min(i, 25)))
1907                End If
1908            Next j
1909        Next i
1910
1911        Dim emisOpPMLDGV(MY, CY) As Double
1912        Dim emisOpPMLDGT1(MY, CY) As Double
1913        Dim emisOpPMLDGT2(MY, CY) As Double
1914        Dim emisOpPMHDGV2b(MY, CY) As Double
1915
1916        Dim emisOpPMLDDV(MY, CY) As Double
1917        Dim emisOpPMLDDV1(MY, CY) As Double
1918        Dim emisOpPMLDDV2(MY, CY) As Double
1919        Dim emisOpPMHDDV2b(MY, CY) As Double
1920
1921        For i = 0 To CY
1922            For j = 0 To MY
1923                If i >= j Then
1924                    emisOpPMLDGV(j, i) = (VMTLDGV(j, i) * emisRatePMLDGV(j, i)) * (10 ^ 6) / ↙
 (454 * 2000 * 1000)
1925                    emisOpPMLDGT1(j, i) = (VMTLDGT1(j, i) * emisRatePMLDGT1(j, i)) * (10 ^ ↙
6) / (454 * 2000 * 1000)
1926                    emisOpPMLDGT2(j, i) = (VMTLDGT2(j, i) * emisRatePMLDGT2(j, i)) * (10 ^ ↙
6) / (454 * 2000 * 1000)
1927                    emisOpPMHDGV2b(j, i) = (VMTHDGV2b(j, i) * emisRatePMHDGV2b(j, i)) * (10 ↙
^ 6) / (454 * 2000 * 1000)
1928
1929                    emisOpPMLDDV(j, i) = (VMTLDDV(j, i) * CAFE.Parameters.pmDiesLDDV(j, Math ↙
.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1930                    emisOpPMLDDV1(j, i) = (VMTLDDV1(j, i) * CAFE.Parameters.pmDiesLDDT1(j, ↙
Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1931                    emisOpPMLDDV2(j, i) = (VMTLDDV2(j, i) * CAFE.Parameters.pmDiesLDDT2(j, ↙
Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1932                    emisOpPMHDDV2b(j, i) = (VMTHDDV2b(j, i) * CAFE.Parameters.pmDiesHDDV2b(j ↙
, Math.Min(i, 25))) * (10 ^ 6) / (454 * 2000 * 1000)
1933                End If
1934            Next
1935        Next
1936
1937        'Total ("Well to Wheel") Particulate Matter (PM) Emissions
1938        Dim emisTotPMLDGV(MY, CY) As Double
1939        Dim emisTotPMLDGT1(MY, CY) As Double
1940        Dim emisTotPMLDGT2(MY, CY) As Double
1941        Dim emisTotPMHDGV2b(MY, CY) As Double
1942
1943        Dim emisTotPMLDDV(MY, CY) As Double
1944        Dim emisTotPMLDDV1(MY, CY) As Double
1945        Dim emisTotPMLDDV2(MY, CY) As Double
1946        Dim emisTotPMHDDV2b(MY, CY) As Double
1947
1948        'TOTALS:
1949        Dim gasEmisTotPMTotal(MY, CY) As Double
1950        Dim diesEmisTotPMTotal(MY, CY) As Double
1951
1952        For i = 0 To CY
1953            For j = 0 To MY
1954                If i >= j Then
1955                    emisTotPMLDGV(j, i) = emisExtrPMLDGV(j, i) + emisTransPMLDGV(j, i) + ↙
emisRefPMLDGV(j, i) + emisTSDPMLDGV(j, i) + emisOpPMLDGV(j, i)
1956                    emisTotPMLDGT1(j, i) = emisExtrPMLDGT1(j, i) + emisTransPMLDGT1(j, i) + ↙
emisRefPMLDGT1(j, i) + emisTSDPMLDGT1(j, i) + emisOpPMLDGT1(j, i)
1957                    emisTotPMLDGT2(j, i) = emisExtrPMLDGT2(j, i) + emisTransPMLDGT2(j, i) + ↙
emisRefPMLDGT2(j, i) + emisTSDPMLDGT2(j, i) + emisOpPMLDGT2(j, i)
```