

Final Research Report
Research Project T1803, Task 36
Multimodal Investment

MULTIMODAL INVESTMENT CHOICE ANALYSIS

VOLUME II: PROGRAM CODE

by

Rhonda Young
Graduate Research Assistant

Jennifer Barnes
Graduate Research Assistant

G. Scott Rutherford
Professor and Chair
Department of Civil and Environmental Engineering
University of Washington, Box 352700
Seattle, Washington 98195

Washington State Transportation Center (TRAC)
University of Washington, Box 354802
University District Building
1107 NE 45th Street, Suite 535
Seattle, Washington 98105-4631

Washington State Department of Transportation
Technical Monitor
Charles Howard, Director
Planning and Policy Office

Prepared for

Washington State Transportation Commission
Department of Transportation
and in cooperation with
U.S. Department of Transportation
Federal Highway Administration

June 2002

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO. WA-RD 547.2	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.	
4. TITLE AND SUBTITLE MULTIMODAL INVESTMENT CHOICE ANALYSIS Volume II: Program Code		5. REPORT DATE June 2002	
		6. PERFORMING ORGANIZATION CODE	
7. AUTHOR(S) Rhonda Young, Jennifer Barnes, G. Scott Rutherford		8. PERFORMING ORGANIZATION REPORT NO.	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Washington State Transportation Center (TRAC) University of Washington, Box 354802 University District Building; 1107 NE 45th Street, Suite 535 Seattle, Washington 98105-4631		10. WORK UNIT NO.	
		11. CONTRACT OR GRANT NO. Agreement T1803, Task 36	
12. SPONSORING AGENCY NAME AND ADDRESS Washington State Department of Transportation Transportation Building, MS 7370 Olympia, Washington 98504-7370 Project Manager: Doug Brodin, (360) 705-7972		13. TYPE OF REPORT AND PERIOD COVERED Final research report	
		14. SPONSORING AGENCY CODE	
15. SUPPLEMENTARY NOTES This study was conducted in cooperation with the U.S. Department of Transportation, Federal Highway Administration.			
16. ABSTRACT <p style="text-align: center;">The Multimodal Investment Choice Analysis (MICA) project is developing a computer-based tool to assist the Washington State Department of Transportation (WSDOT), the Washington State Transportation Commission, and the Washington State Legislature in making state transportation funding decisions.</p> <p style="text-align: center;">The function of MICA is to summarize the multimodal budgetary tradeoffs that will result from varying funding allocation and priority scenarios. MICA's methodology is based on benefit-cost and goal achievement analyses.</p> <p style="text-align: center;">The project has completed the first phase of research. The purpose of this two-volume report is to document the analysis methodology contained within the MICA program, as well as general findings from the first phase of this research effort. This document is also designed to serve as a User's Manual for operating the MICA program. While challenges still exist, the results of the Phase I research effort indicate that development of a multimodal analysis tool is feasible.</p>			
17. KEY WORDS Statewide programming, prioritization, multimodal		18. DISTRIBUTION STATEMENT No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22616	
19. SECURITY CLASSIF. (of this report) None	20. SECURITY CLASSIF. (of this page) None	21. NO. OF PAGES	22. PRICE

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Washington State Transportation Commission, Department of Transportation, or the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

TABLE OF CONTENTS

Program Code	1
Program Code for Ferry Projects	1
Program Code for Ferry Preservation Projects	1
Program Code for Ferry Construction Projects	10
Program Code for Highway Improvement Projects.....	21
Climbing Lane	21
General Purpose Lane	36
High Occupancy Vehicle Lane	51
Interchange.....	68
Intersection.....	81
Park and Ride.....	95
Two-Way-Left-Turn Lane	106
Program Code for Highway Preservation Projects	122
Pavement Preservation Projects	122
Structure Preservation Projects.....	128
Program Code for Highway Safety Projects	136
(Program code is the same for all project types).....	136
Program Code for Intelligent Transportation Systems	144
IDAS Calculations	144
SCRITS Calculations.....	152
Program Code for Non-Motorized Projects.....	193
Program Code for Rail Projects.....	203
Freight Car Purchase	203
Grade Separation / Crossing Improvement.....	215
Modal Connection Improvement	224
Passenger Trainset Purchase	235
Station Improvement	245

TABLE OF CONTENTS

Track Improvement	255
Program Code for Transit Projects	268
STEAM Calculations	268
SPASM Calculations	280
Program Code for Transportation Demand Management	293
Areawide Programs	293
Commute Trip Reduction Support.....	303
Program Code for Scenarios and Optimization.....	314

Program Code

Program Code for Ferry Projects

Program Code for Ferry Preservation Projects

The following code is for ferry terminal preservation projects. Vessel preservation projects are similar except that only one route would be impacted by a service failure while terminals may have multiple routes. Where differences occur they have been noted in the code by bold text within brackets.

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim a_rst As DAO.Recordset

Dim m_qryName As String

Dim m_calcsComplete As Boolean

Dim m_Values As ValueCollection

'Variables used in intermediate calculations (variants)

Dim m_VitalScr

Dim m_NonVitalScr

Dim m_Prob(1 To 5)

Dim m_VMT_PC(1 To 5)

Dim m_Trips_PC(1 To 5)

'Variables to hold the calculated values (variants)

Dim m_Week_Trips_NS

Dim m_Weekend_Trips_NS

Dim m_Week_Wait

Dim m_Weekend_Wait

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_User_Transfer

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Terminal_Cost

Dim m_BCR

Dim m_WSDOT_BCR

'Variables to hold the outcome objective scores (variants)

Dim m_Sys_OM

Dim m_Sys_Pres

Dim m_Sp_Needs

Dim m_Cong_Rel

Dim m_Trav_Opt

Dim m_Seamless

```

Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource

```

```

'Speed Assumed for lookup table
Private Const AUTOSPEED = 50

```

```

Private Sub Class_Initialize()
    m_qryName = "fry_calc_Preservation_Terminal"
    m_calcsComplete = False
End Sub

```

```

Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean

```

```

    Dim atype As String
    Dim qryDef As DAO.QueryDef

```

```

    Set m_dbs = CurrentDb

```

```

    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid

```

```

    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then

```

```

        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection

```

```

        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If

```

```

        'Open up a assumption recordset - depend on if in MICA or not
        Set qryDef = m_dbs.QueryDefs(atype)
        qryDef.Parameters!asmptnID = pid

```

```

        ' Set recordset to new values.
        Set a_rst = qryDef.OpenRecordset

```

```

        'Calculate all of the values for the project type
        CalculateBenefits
        CalculateOutObjScores

```

```

        'Check to see if the calcs are done for this project and set input status accordingly
        If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
        m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

```



```

    Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
[The following calculations are repeated for each route served by the terminal. For vessel preservation projects they are only done once.]
    On Error Resume Next

    'Calculate the Vessel Probability Failures
    For i = 1 To 5
        m_Prob(i) = (((100 - m_rst("CR_Vital_BC" & i)) / 100) - ((100 - m_rst("CR_Vital_PC" & i)) / 100))
    Next

    While Not m_rst.EOF
        'Calculations for weighted weekday percentages for trip purpose and boarding mode
        Per_Peak = m_rst("Per_Peak")

        Week_WorkPer = m_rst("Work_Peak") * (2 * Per_Peak) + (m_rst("Work_Non") * (1 - (2 * Per_Peak)))
        Week_MedPer = m_rst("Med_Peak") * (2 * Per_Peak) + (m_rst("Med_Non") * (1 - (2 * Per_Peak)))
        Week_SocialPer = m_rst("Social_Peak") * (2 * Per_Peak) + (m_rst("Social_Non") * (1 - (2 * Per_Peak)))
        Week_Veh = m_rst("Peak_Veh") * (2 * Per_Peak) + (m_rst("Non_Veh") * (1 - (2 * Per_Peak)))
        Week_Walk = m_rst("Peak_Walk") * (2 * Per_Peak) + (m_rst("Non_Walk") * (1 - (2 * Per_Peak)))

        'Calculations for percentage of riders likely to drive around during service failure (NS=no service)
        Week_TotalPer_NS = Week_WorkPer + 0.5 * Week_MedPer + 0.1 * Week_SocialPer
        Weekend_TotalPer_NS = m_rst("Work_Sun") + 0.5 * m_rst("Med_Sun") + _
            0.1 * m_rst("Social_Sun")

        'Calculations for number of drive around trips during service failure for two-year period
        Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
        Week_Rider = m_rst("Week_Rider")
        Weekend_Rider = m_rst("Weekend_Rider")
        [Vessel Preservation Projects divide the Week_Rider and Weekend_Rider number by the number of vessels serving the route, Num_Vessel]

        m_Week_Trips_NS = m_Week_Trips_NS + (Week_Rider * Week_TotalPer_NS * Ann_Daily_Benefit)
        m_Weekend_Trips_NS = m_Weekend_Trips_NS + (Weekend_Rider * Weekend_TotalPer_NS * (365 - Ann_Daily_Benefit))

        'Boat Wait Calculations
        Peak_Wait = ((m_rst("Peak_5") * 5) + (m_rst("Peak_20") * 20) + _
            (m_rst("Peak_45") * 45) + (m_rst("Peak_75") * 75))
        Non_Wait = ((m_rst("Non_5") * 5) + (m_rst("Non_20") * 20) + _
            (m_rst("Non_45") * 45) + (m_rst("Non_75") * 75))
    End While
End Sub

```

```

m_Weekend_Wait = m_Weekend_Wait + (((m_rst("Sun_5") * 5) + (m_rst("Sun_20") * 20) + _
(m_rst("Sun_45") * 45) + (m_rst("Sun_75") * 75)))
m_Week_Wait = m_Week_Wait + (Peak_Wait * (2 * Per_Peak) + _
Non_Wait * (1 - (2 * Per_Peak)))

```

'Travel Time Calculations

```

Discount_Rate = a_rst("Discount_Rate")
Week_Trips = Week_Rider * Ann_Daily_Benefit
Weekend_Trips = Weekend_Rider * (365 - Ann_Daily_Benefit)
For i = 1 To 5
  m_Trips_PC(i) = (Week_Trips + Weekend_Trips) * m_Prob(i)
Next

TT_Min_OutVeh = (Week_Wait * Week_Trips_NS) + (Weekend_Wait * Weekend_Trips_NS)
TT_Min_InVeh = ((m_rst("TT_NS") - m_rst("TT_BC")) * m_Week_Trips_NS) + _
((m_rst("TT_NS") - m_rst("TT_BC")) * m_Weekend_Trips_NS)
For i = 1 To 5
  TT_Min_IV = m_Prob(i) * (TT_Min_InVeh)
  TT_Min_OV = m_Prob(i) * (TT_Min_OutVeh)
  tt_ben_i = (TT_Min_IV / 60 * a_rst("Percent_TV_InVeh") * a_rst("Time_Value_Veh")) - _
(TT_Min_OV / 60 * a_rst("Percent_TV_OutVeh") * a_rst("Time_Value_Veh"))
  m_TT_Min = m_TT_Min + (TT_Min_IV - TT_Min_OV)
  m_TT_Ben = m_TT_Ben + (tt_ben_i / ((1 + Discount_Rate) ^ (2 * i)))
Next

```

'Operating Cost Calculations

```

For i = 1 To 5
  m_VMT_PC(i) = m_Prob(i) * (TT_Min_InVeh) / 60 * 50
Next

```

'User Cost Benefit Calculations

```

Veh_OpCost_Full = a_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = a_rst("Veh_OpCost_Direct")
For i = 1 To 5
  If a_rst("Full_Cost") Then
    User_Ben_PC = m_VMT_PC(i) * Veh_OpCost_Full
  Else
    User_Ben_PC = m_VMT_PC(i) * Veh_OpCost_Direct
  End If
  'User Benefit NPV Calculation
  m_User_Ben = m_User_Ben + (User_Ben_PC / ((1 + Discount_Rate) ^ (2 * i)))
Next

```

'Air Pollution Calculations

'Emissions Calculations for Drive Around Trips

```

Per_Cold_Auto = a_rst("Per_Cold_Auto")
CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)
If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If
CO_Cold_Auto = a_rst("CO_Cold_Auto")
CO_Ton_Cost = a_rst("CO_Ton_Cost")
VOC_Rate_Auto = a_rst("VOC_Rate_Auto")
VOC_Cold_Auto = a_rst("VOC_Cold_Auto")
VOCTon_Cost = a_rst("VOCTon_Cost")
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
NOX_Cold_Auto = a_rst("NOX_Cold_Auto")

```

```

NOXTon_Cost = a_rst("NOXTon_Cost")
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")
PM10_Cold_Auto = a_rst("PM10_Cold_Auto")
PM10Ton_Cost = a_rst("PM10Ton_Cost")

For i = 1 To 5
  CO_Tons_i = ((m_VMT_PC(i) * CO_Rate_Auto) + _
    (m_Trips_PC(i) * Per_Cold_Auto * CO_Cold_Auto)) * _
    (1 / 1000) * (0.9842 / 1000)
  VOC_Tons_i = ((m_VMT_PC(i) * VOC_Rate_Auto) + _
    (m_Trips_PC(i) * Per_Cold_Auto * VOC_Cold_Auto)) * _
    (1 / 1000) * (0.9842 / 1000)
  NOX_Tons_i = ((m_VMT_PC(i) * NOX_Rate_Auto) + _
    (m_Trips_PC(i) * Per_Cold_Auto * NOX_Cold_Auto)) * _
    (1 / 1000) * (0.9842 / 1000)
  PM10_Tons_i = ((m_VMT_PC(i) * PM10_Rate_Auto) + _
    (m_Trips_PC(i) * Per_Cold_Auto * PM10_Cold_Auto)) * _
    (1 / 1000) * (0.9842 / 1000)
  'Emmision Sums
  m_CO_Tons = m_CO_Tons - CO_Tons_i
  m_VOC_Tons = m_VOC_Tons - VOC_Tons_i
  m_NOX_Tons = m_NOX_Tons - NOX_Tons_i
  m_PM10_Tons = m_PM10_Tons - PM10_Tons_i
  'Emission Benefit Calculations
  Env_Ben_i = CO_Tons_i * COton_Cost + VOC_Tons_i * VOCTon_Cost + _
    NOX_Tons_i * NOXTon_Cost + PM10_Tons_i * PM10Ton_Cost
  'Emission NPV Calculations
  m_Env_Ben = m_Env_Ben + (Env_Ben_i / ((1 + Discount_Rate) ^ (2 * i)))
Next

'Lost Fare Revenue Calculations
Fare_Veh = m_rst("Fare_Veh")
Fare_Walk = m_rst("Fare_Walk")
Sun_Walk = m_rst("Sun_Walk")
Sun_Veh = m_rst("Sun_Veh")

For i = 1 To 5
  User_Transfer_I = (((Week_Rider * Week_Veh * Fare_Veh) + _
    (Week_Rider * Week_Walk * Fare_Walk * 0.5)) * Ann_Daily_Benefit) + _
    (((Weekend_Rider * Sun_Veh * Fare_Veh) + _
    (Weekend_Rider * Sun_Walk * Fare_Walk * 0.5)) * (365 - Ann_Daily_Benefit)) * _
    (m_Prob(i))
  'Revenue NPV Calculations
  m_User_Transfer = m_User_Transfer + (User_Transfer_I / ((1 + Discount_Rate) ^ (2 * i)))
Next

'Safety Calculations
Fat_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_auto)
If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
End If
Inj_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Injury, veht_auto)
If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
End If
Prop_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Property, veht_auto)
If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
End If

For i = 1 To 5

```

```

Fatalitiy_i = m_VMT_PC(i) * Fat_Rate_Auto / 100000000
Injury_i = m_VMT_PC(i) * Inj_Rate_Auto / 1000000
Property_i = m_VMT_PC(i) * Prop_Rate_Auto / 1000000
'Safety Sums
  m_Fatality = m_Fatality - Fatalitiy_i
  m_Injury = m_Injury - Injury_i
  m_Property = m_Property - Property_i
'Safety Benefit Calculations
Safety_Ben_i = Fatalitiy_i * a_rst("Fatality_Cost") + Injury_i * a_rst("Evident_Cost") + _
  Property_i * a_rst("PDO_Cost")
'Safety NPV Calculations
m_Safety_Ben = m_Safety_Ben + (Safety_Ben_i / ((1 + Discount_Rate) ^ (2 * i)))
Next

'End of iterative loop time to store final values
[End of iterative loop for analyzing multiple routes served by terminal.]
m_rst.MoveNext
Wend
m_rst.MoveFirst

'Set summed module level values (to be displayed)
m_Values.Add m_Week_Trips_NS, "WEEK_TRIPS_NS"
m_Values.Add m_Weekend_Trips_NS, "WEEKEND_TRIPS_NS"
m_Values.Add m_Week_Wait, "WEEK_WAIT"
m_Values.Add m_Weekend_Wait, "WEEKEND_WAIT"
m_Values.Add m_TT_Min, "TT_MIN"
m_Values.Add m_TT_Ben, "TT_BEN"
m_Values.Add m_User_Ben, "USER_BEN"
m_Values.Add m_User_Transfer, "USER_TRANSFER"
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"
m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"
m_Values.Add m_PM10_Tons, "PM10_TONS"
m_Values.Add m_Env_Ben, "ENV_BEN"

'Calculate economic analysis factors
PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / (Discount_Rate * (1 +
Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / ((1 + Discount_Rate) ^ Forecast_Period))
PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Cost Calculations
'Capital Cost
For i = 1 To 5
  m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i -
1)))
  m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i -
1)))
  m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i -
1)))
  m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i -
1)))
  m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i -
1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

```

```

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + m_Other1_Cost + m_Other2_Cost +
m_Other3_Cost

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
'Environmental Retrofit Costs

For i = 1 To 5
    m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * 1 / ((1 + Discount_Rate) ^
(2 * i - 1)))
    m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * 1 / ((1 + Discount_Rate)
^ (2 * i - 1)))
    m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * 1 / ((1 +
Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * a_rst("fishbarrier_bc")
m_StormWater_Ben = m_StormWater_Cap * a_rst("stormwater_bc")
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * a_rst("noisebarrier_bc")

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben + m_EnvRetrofit_Ben
m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)

'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"

```

```

m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

'First create all intermediate values used in the following calcs

'Calculate the Vital/Non-Vital Scores

For i = 1 To 5

```

tmp1 = m_rst("CR_Vital_BC" & i).Value
tmp2 = m_rst("CR_NonVital_BC" & i).Value
tmp3 = m_rst("CR_Vital_PC" & i).Value
tmp4 = m_rst("CR_NonVital_PC" & i).Value

```

```

m_VitalScr = m_VitalScr + ((tmp3 - tmp1) / 5)
m_NonVitalScr = m_NonVitalScr + ((tmp4 - tmp2) / 5)

```

Next

'Set to integers and If values are less than one set them to zero

If m_VitalScr < 0 Then m_VitalScr = 0

If m_NonVitalScr < 0 Then m_NonVitalScr = 0

'Calculation for the System Operation and Maintenance

```

m_Sys_OM = (m_rst("Q1A") * 34 + (0.33 * ((m_VitalScr * 2 + m_NonVitalScr) / 3))) _
+ 0 + (33 * m_rst("Q1D"))
m_Values.Add m_Sys_OM, "SYS_OM"

```

'Calculation for the System Preservation

```

m_Sys_Pres = 50 + (50 - ((m_VitalScr * 2 + m_NonVitalScr) / 3 * 0.5))
m_Values.Add m_Sys_Pres, "SYS_PRES"

```

'Calculation for the Special Needs Transportation

```

m_Sp_Needs = 100 * m_rst("Q3A")
m_Values.Add m_Sp_Needs, "SP_NEEDS"

```

'Calculation for the Congestion Relief

If m_rst("WTP_Corridor") = Yes Then

m_Cong_Rel = 50

Else

m_Cong_Rel = 0

End If

m_Values.Add m_Cong_Rel, "CONG_REL"

'Calculation for Increased Travel Options

If m_rst("Ferry_Class") = "Passenger Only Ferry" Then

m_Trav_Opt = 33

Else

m_Trav_Opt = 0

End If

m_Values.Add m_Trav_Opt, "TRAV_OPT"

'Calculation for Seamless Connections

```

m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)

```

```

    m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
    m_Safety = (m_rst("Q7A") * 50) + (m_rst("Q7B") * 50)
    m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
    'The -1 Multiplication Corrects the negative sign introduced for true
    m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E")))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") > 0 Then
        m_Collab = 50
    Else
        m_Collab = 0
    End If
    m_Collab = m_Collab + (50 * m_rst("Q10A"))
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    If m_rst("Q14B") = 1 Then
        m_Air_Qual = 50 + (50 * m_rst("Q14A"))
    Else
        m_Air_Qual = 0
    End If
    m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close

```

```

m_rst.Close
m_dbs.Close
Set a_rst = Nothing
Set m_rst = Nothing
Set m_dbs = Nothing
Set m_Values = Nothing
End Sub

```

Program Code for Ferry Construction Projects

The following code is for ferry terminal construction projects. Vessel construction projects are identical.

Option Compare Database

'Variables for the class level database objects

```

Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection

```

'Variables used in intermediate calculations (variants)

```

Dim m_VitalScr
Dim m_NonVitalScr
Dim m_Prob(1 To 5)
Dim m_VMT_PC(1 To 5)

```

'Variables to hold the calculated values (variants)

```

Dim m_TT_Min
Dim m_TT_Ben
Dim m_User_Ben
Dim m_User_Transfer
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR

```

'Variables to hold the outcome objective scores (variants)

```

Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight

```



```

Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource

'Speed Assumed for lookup table
Private Const AUTOSPEED = 50

Private Sub Class_Initialize()
    m_qryName = "fry_calc_Construction_Terminal"
    m_calcsComplete = False
End Sub

Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean

    Dim atype As String
    Dim qryDef As DAO.QueryDef

    Set m_dbs = CurrentDb

    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid

    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then

        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection

        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If

        'Open up a assumption recordset - depend on if in MICA or not
        Set qryDef = m_dbs.QueryDefs(atype)
        qryDef.Parameters!asmptnID = pid

        ' Set recordset to new values.
        Set a_rst = qryDef.OpenRecordset

        'Calculate all of the values for the project type
        CalculateBenefits
        CalculateOutObjScores

        'Check to see if the calcs are done for this project and set input status accordingly
        If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
        m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

        Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

    End If

    'Close the Querydef object

```

```

qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Travel Time Calculations:
'Travel Time Savings per Rider
    Rider_WkDy_BCI = m_rst("Rider_WkDy_BCI")
    Rider_WkDy_BCF = m_rst("Rider_WkDy_BCF")
    Rider_WkDy_PCI = m_rst("Rider_WkDy_PCI")
    Rider_WkDy_PCF = m_rst("Rider_WkDy_PCF")
    Rider_WkEd_BCI = m_rst("Rider_WkEd_BCI")
    Rider_WkEd_BCF = m_rst("Rider_WkEd_BCF")
    Rider_WkEd_PCI = m_rst("Rider_WkEd_PCI")
    Rider_WkEd_PCF = m_rst("Rider_WkEd_PCF")
    AutoTT_WkDy_BCI = m_rst("AutoTT_WkDy_BCI") / Rider_WkDy_BCI
    AutoTT_WkDy_BCF = m_rst("AutoTT_WkDy_BCF") / Rider_WkDy_BCF
    AutoTT_WkDy_PCI = m_rst("AutoTT_WkDy_PCI") / Rider_WkDy_PCI
    AutoTT_WkDy_PCF = m_rst("AutoTT_WkDy_PCF") / Rider_WkDy_PCF
    WaitTT_WkDy_BCI = m_rst("WaitTT_WkDy_BCI") / Rider_WkDy_BCI
    WaitTT_WkDy_BCF = m_rst("WaitTT_WkDy_BCF") / Rider_WkDy_BCF
    WaitTT_WkDy_PCI = m_rst("WaitTT_WkDy_PCI") / Rider_WkDy_PCI
    WaitTT_WkDy_PCF = m_rst("WaitTT_WkDy_PCF") / Rider_WkDy_PCF
    FerryTT_WkDy_BCI = m_rst("FerryTT_WkDy_BCI") / Rider_WkDy_BCI
    FerryTT_WkDy_BCF = m_rst("FerryTT_WkDy_BCF") / Rider_WkDy_BCF
    FerryTT_WkDy_PCI = m_rst("FerryTT_WkDy_PCI") / Rider_WkDy_PCI
    FerryTT_WkDy_PCF = m_rst("FerryTT_WkDy_PCF") / Rider_WkDy_PCF
    AutoTT_WkEd_BCI = m_rst("AutoTT_WkEd_BCI") / Rider_WkEd_BCI
    AutoTT_WkEd_BCF = m_rst("AutoTT_WkEd_BCF") / Rider_WkEd_BCF
    AutoTT_WkEd_PCI = m_rst("AutoTT_WkEd_PCI") / Rider_WkEd_PCI
    AutoTT_WkEd_PCF = m_rst("AutoTT_WkEd_PCF") / Rider_WkEd_PCF
    WaitTT_WkEd_BCI = m_rst("WaitTT_WkEd_BCI") / Rider_WkEd_BCI
    WaitTT_WkEd_BCF = m_rst("WaitTT_WkEd_BCF") / Rider_WkEd_BCF
    WaitTT_WkEd_PCI = m_rst("WaitTT_WkEd_PCI") / Rider_WkEd_PCI
    WaitTT_WkEd_PCF = m_rst("WaitTT_WkEd_PCF") / Rider_WkEd_PCF
    FerryTT_WkEd_BCI = m_rst("FerryTT_WkEd_BCI") / Rider_WkEd_BCI
    FerryTT_WkEd_BCF = m_rst("FerryTT_WkEd_BCF") / Rider_WkEd_BCF
    FerryTT_WkEd_PCI = m_rst("FerryTT_WkEd_PCI") / Rider_WkEd_PCI
    FerryTT_WkEd_PCF = m_rst("FerryTT_WkEd_PCF") / Rider_WkEd_PCF

'Induced Ridership Values
    Induced_WkDy_I = Rider_WkDy_PCI - Rider_WkDy_BCI
    Induced_WkEd_I = Rider_WkEd_PCI - Rider_WkEd_BCI
    Induced_WkDy_F = Rider_WkDy_PCF - Rider_WkDy_BCF
    Induced_WkEd_F = Rider_WkEd_PCF - Rider_WkEd_BCF

'Travel Time Benefits in Minutes -Weekday Yearly Total (negative = more travel time)
    Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
    AutoTT_WkDyMin_I = -((Rider_WkDy_BCI * (AutoTT_WkDy_PCI - AutoTT_WkDy_BCI)) + _
        (Induced_WkDy_I * 0.5 * (AutoTT_WkDy_PCI - AutoTT_WkDy_BCI))) * _

```

```

Ann_Daily_Benefit
AutoTT_WkDyMin_F = -((Rider_WkDy_BCI * (AutoTT_WkDy_PCF - AutoTT_WkDy_BCF)) +
(Induced_WkDy_F * 0.5 * (AutoTT_WkDy_PCF - AutoTT_WkDy_BCF))) *
Ann_Daily_Benefit
WaitTT_WkDyMin_I = -((Rider_WkDy_BCI * (WaitTT_WkDy_PCI - WaitTT_WkDy_BCI)) +
(Induced_WkDy_I * 0.5 * (WaitTT_WkDy_PCI - WaitTT_WkDy_BCI))) *
Ann_Daily_Benefit
WaitTT_WkDyMin_F = -((Rider_WkDy_BCI * (WaitTT_WkDy_PCF - WaitTT_WkDy_BCF)) +
(Induced_WkDy_F * 0.5 * (WaitTT_WkDy_PCF - WaitTT_WkDy_BCF))) *
Ann_Daily_Benefit
FerryTT_WkDyMin_I = -((Rider_WkDy_BCI * (FerryTT_WkDy_PCI - FerryTT_WkDy_BCI)) +
(Induced_WkDy_I * 0.5 * (FerryTT_WkDy_PCI - FerryTT_WkDy_BCI))) *
Ann_Daily_Benefit
FerryTT_WkDyMin_F = -((Rider_WkDy_BCI * (FerryTT_WkDy_PCF - FerryTT_WkDy_BCF)) +
(Induced_WkDy_F * 0.5 * (FerryTT_WkDy_PCF - FerryTT_WkDy_BCF))) *
Ann_Daily_Benefit
TT_Min_WkDy_I = AutoTT_WkDyMin_I + WaitTT_WkDyMin_I + FerryTT_WkDyMin_I
TT_Min_WkDy_F = AutoTT_WkDyMin_F + WaitTT_WkDyMin_F + FerryTT_WkDyMin_F

```

'Travel Time Benefits in Minutes -Weekend Yearly Total

```

AutoTT_WkEdMin_I = -((Rider_WkEd_BCI * (AutoTT_WkEd_PCI - AutoTT_WkEd_BCI)) +
(Induced_WkEd_I * 0.5 * (AutoTT_WkEd_PCI - AutoTT_WkEd_BCI))) *
(365 - Ann_Daily_Benefit)
AutoTT_WkEdMin_F = -((Rider_WkEd_BCI * (AutoTT_WkEd_PCF - AutoTT_WkEd_BCF)) +
(Induced_WkEd_F * 0.5 * (AutoTT_WkEd_PCF - AutoTT_WkEd_BCF))) *
(365 - Ann_Daily_Benefit)
WaitTT_WkEdMin_I = -((Rider_WkEd_BCI * (WaitTT_WkEd_PCI - WaitTT_WkEd_BCI)) +
(Induced_WkEd_I * 0.5 * (WaitTT_WkEd_PCI - WaitTT_WkEd_BCI))) *
(365 - Ann_Daily_Benefit)
WaitTT_WkEdMin_F = -((Rider_WkEd_BCI * (WaitTT_WkEd_PCF - WaitTT_WkEd_BCF)) +
(Induced_WkEd_F * 0.5 * (WaitTT_WkEd_PCF - WaitTT_WkEd_BCF))) *
(365 - Ann_Daily_Benefit)
FerryTT_WkEdMin_I = -((Rider_WkEd_BCI * (FerryTT_WkEd_PCI - FerryTT_WkEd_BCI)) +
(Induced_WkEd_I * 0.5 * (FerryTT_WkEd_PCI - FerryTT_WkEd_BCI))) *
(365 - Ann_Daily_Benefit)
FerryTT_WkEdMin_F = -((Rider_WkEd_BCI * (FerryTT_WkEd_PCF - FerryTT_WkEd_BCF)) +
(Induced_WkEd_F * 0.5 * (FerryTT_WkEd_PCF - FerryTT_WkEd_BCF))) *
(365 - Ann_Daily_Benefit)
TT_Min_WkEd_I = AutoTT_WkEdMin_I + WaitTT_WkEdMin_I + FerryTT_WkEdMin_I
TT_Min_WkEd_F = AutoTT_WkEdMin_F + WaitTT_WkEdMin_F + FerryTT_WkEdMin_F

```

'Travel Time Benefits in Minutes Total

```

Percent_TV_InVeh = a_rst("Percent_TV_InVeh")
Percent_TV_OutVeh = a_rst("Percent_TV_OutVeh")
Time_Value_Veh = a_rst("Time_Value_Veh")
Discount_Rate = a_rst("Discount_Rate")
TT_Min_I = TT_Min_WkDy_I + TT_Min_WkEd_I
TT_Min_F = TT_Min_WkDy_F + TT_Min_WkEd_F
N = m_rst("Fore_Year") - m_rst("Init_Year")
LogTT_Min = ReturnLog_val1_div_val2(TT_Min_F, TT_Min_I)
If Not (IsNull(LogTT_Min)) And LogTT_Min <> 0 And IsNumeric(N) And N > 0 Then
  NPVF_Min = (Exp(LogTT_Min) - 1) /
  ((LogTT_Min / N))
Else
  NPVF_Min = N
End If
m_TT_Min = TT_Min_I * NPVF_Min
m_FrTT_Min = 0
'Set module level values (to be displayed)
m_Values.Add m_TT_Min, "TT_MIN"

```

'Travel Time Benefits in Dollars

```

TT_Ben_Auto_I = (AutoTT_WkEdMin_I + AutoTT_WkDyMin_I) / 60 * _
    (Percent_TV_InVeh * Time_Value_Veh)
TT_Ben_Wait_I = (WaitTT_WkEdMin_I + WaitTT_WkDyMin_I) / 60 * _
    (Percent_TV_OutVeh * Time_Value_Veh)
TT_Ben_Ferry_I = (FerryTT_WkEdMin_I + FerryTT_WkDyMin_I) / 60 * _
    (Percent_TV_InVeh * Time_Value_Veh)
tt_ben_i = TT_Ben_Auto_I + TT_Ben_Wait_I + TT_Ben_Ferry_I
TT_Ben_Auto_F = (AutoTT_WkEdMin_F + AutoTT_WkDyMin_F) / 60 * _
    (Percent_TV_InVeh * Time_Value_Veh)
TT_Ben_Wait_F = (WaitTT_WkEdMin_F + WaitTT_WkDyMin_F) / 60 * _
    (Percent_TV_OutVeh * Time_Value_Veh)
TT_Ben_Ferry_F = (FerryTT_WkEdMin_F + FerryTT_WkDyMin_F) / 60 * _
    (Percent_TV_InVeh * Time_Value_Veh)
tt_ben_f = TT_Ben_Auto_F + TT_Ben_Wait_F + TT_Ben_Ferry_F
'Null cannot be used in exp() or log()
LogTT_Ben = ReturnLog_val1_div_val2(tt_ben_f, tt_ben_i)
If Not (IsNull(LogTT_Ben)) And LogTT_Ben <> 0 And IsNumeric(N) And N > 0 Then
    NPVF_TTBen = (Exp(((LogTT_Ben / N) - Discount_Rate) * N) - 1) / _
        ((LogTT_Ben / N) - Discount_Rate)
Else
    NPVF_TTBen = (Exp(-Discount_Rate * N) - 1) / (-Discount_Rate)
End If
m_TT_Ben = tt_ben_i * NPVF_TTBen
m_FrTT_Ben = 0
'Set module level values (to be displayed)
m_Values.Add m_TT_Ben, "TT_BEN"

'Operating Cost Calculations
VMT_I = -(((m_rst("VMT_WkEd_PCI") - m_rst("VMT_WkEd_BCI")) * (365 - Ann_Daily_Benefit)) + _
    ((m_rst("VMT_WkDy_PCI") - m_rst("VMT_WkDy_BCI")) * Ann_Daily_Benefit))
VMT_F = -(((m_rst("VMT_WkEd_PCF") - m_rst("VMT_WkEd_BCF")) * (365 - Ann_Daily_Benefit)) + _
    ((m_rst("VMT_WkDy_PCF") - m_rst("VMT_WkDy_BCF")) * Ann_Daily_Benefit))
LogNPVF_VMT = ReturnLog_val1_div_val2(VMT_F, VMT_I)
If Not (IsNull(LogNPVF_VMT)) And LogNPVF_VMT <> 0 And IsNumeric(N) And N > 0 Then
    NPVF_VMT = (Exp(LogNPVF_VMT - 1) / (LogNPVF_VMT / N))
Else
    NPVF_VMT = N
End If
VMT_Tot = VMT_I * NPVF_VMT

'User Cost Benefit Calculations
If a_rst("Full_Cost") Then
    UserBen_I = VMT_I * a_rst("Veh_OpCost_Full")
    UserBen_F = VMT_F * a_rst("Veh_OpCost_Full")
Else
    UserBen_I = VMT_I * a_rst("Veh_OpCost_Direct")
    UserBen_F = VMT_F * a_rst("Veh_OpCost_Direct")
End If
LogUserBen = ReturnLog_val1_div_val2(UserBen_F, UserBen_I)
If Not (IsNull(LogUserBen)) And LogUserBen <> 0 And IsNumeric(N) And N > 0 Then
    NPVF_UCBen = (Exp(((LogUserBen / N) - Discount_Rate) * N) - 1) / _
        ((LogUserBen / N) - Discount_Rate)
Else
    NPVF_UCBen = (Exp(-Discount_Rate * N) - 1) / (-Discount_Rate)
End If
m_User_Ben = UserBen_I * NPVF_UCBen
'Set module level values (to be displayed)
m_Values.Add m_User_Ben, "USER_BEN"

'Air Pollution Emissions Calculations (does not yet consider ferry emissions)

'Set assumption variables

```

```

CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)
If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If
COTon_Cost = a_rst("COTon_Cost")
VOC_Rate_Auto = a_rst("VOC_Rate_Auto")
VOCTon_Cost = a_rst("VOCTon_Cost")
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
NOXTon_Cost = a_rst("NOXTon_Cost")
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")
PM10Ton_Cost = a_rst("PM10Ton_Cost")

m_CO_Tons = VMT_I * NPVF_VMT * CO_Rate_Auto / 907180
m_VOC_Tons = VMT_I * NPVF_VMT * VOC_Rate_Auto / 907180
m_NOX_Tons = VMT_I * NPVF_VMT * NOX_Rate_Auto / 907180
m_PM10_Tons = VMT_I * NPVF_VMT * PM10_Rate_Auto / 907180
'Set module level values (to be displayed)
    m_Values.Add m_CO_Tons, "CO_TONS"
    m_Values.Add m_VOC_Tons, "VOC_TONS"
    m_Values.Add m_NOX_Tons, "NOX_TONS"
    m_Values.Add m_PM10_Tons, "PM10_TONS"

'Emissions Benefit Calculations
If Not (IsNull(LogNPVF_VMT)) And LogNPVF_VMT <> 0 And IsNumeric(N) And N > 0 Then
    NPVF_EnvBen = (Exp(((LogNPVF_VMT / N) - Discount_Rate) * N) - 1) / _
                ((LogNPVF_VMT / N) - Discount_Rate)
Else
    NPVF_EnvBen = (Exp(-Discount_Rate * N) - 1) / (-Discount_Rate)
End If
CO_Ben = VMT_I * CO_Rate_Auto * COTon_Cost * NPVF_EnvBen / 907180
VOC_Ben = VMT_I * VOC_Rate_Auto * VOCTon_Cost * NPVF_EnvBen / 907180
NOX_Ben = VMT_I * NOX_Rate_Auto * NOXTon_Cost * NPVF_EnvBen / 907180
PM10_Ben = VMT_I * PM10_Rate_Auto * PM10Ton_Cost * NPVF_EnvBen / 907180
m_Env_Ben = CO_Ben + VOC_Ben + NOX_Ben + PM10_Ben
'Set module level values (to be displayed)
    m_Values.Add m_Env_Ben, "ENV_BEN"

'Fare Revenue Calculations
Fare_BCI = m_rst("Fare_BCI")
Fare_PCI = m_rst("Fare_PCI")
User_Transfer_WkDy_I = (Rider_WkDy_BCI * (Fare_PCI - Fare_BCI) + _
                    (Induced_WkDy_I * Fare_PCI)) * Ann_Daily_Benefit
User_Transfer_WkEd_I = (Rider_WkEd_BCI * (Fare_PCI - Fare_BCI) + _
                    (Induced_WkEd_I * Fare_PCI)) * (365 - Ann_Daily_Benefit)
User_Transfer_I = User_Transfer_WkDy_I + User_Transfer_WkEd_I
User_Transfer_WkDy_F = (Rider_WkDy_BCF * (Fare_PCI - Fare_BCI) + _
                    (Induced_WkDy_F * Fare_PCI)) * Ann_Daily_Benefit
User_Transfer_WkEd_F = (Rider_WkEd_BCF * (Fare_PCI - Fare_BCI) + _
                    (Induced_WkEd_F * Fare_PCI)) * (365 - Ann_Daily_Benefit)
User_Transfer_F = User_Transfer_WkDy_F + User_Transfer_WkEd_F
LogUserTrans = ReturnLog_val1_div_val2(User_Transfer_F, User_Transfer_I)
If Not (IsNull(LogUserTrans)) And LogUserTrans <> 0 And IsNumeric(N) And N > 0 Then
    NPVF_TransBen = (Exp(((LogUserTrans / N) - Discount_Rate) * N) - 1) / _
                ((LogUserTrans / N) - Discount_Rate)
Else
    NPVF_TransBen = (Exp(-Discount_Rate * N) - 1) / (-Discount_Rate)
End If
m_User_Transfer = User_Transfer_I * NPVF_TransBen
'Set module level values (to be displayed)

```

```

m_Values.Add m_User_Transfer, "USER_TRANSFER"

'Safety & Accident Calculations (does not yet include ferry accident rates)
'Set assumption variables
Fat_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_auto)
If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
End If
Inj_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Injury, veht_auto)
If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
End If
Prop_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Property, veht_auto)
If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
End If

m_Fatality = VMT_I * NPVF_VMT * Fat_Rate_Auto / 100000000
m_Injury = VMT_I * NPVF_VMT * Inj_Rate_Auto / 1000000
m_Property = VMT_I * NPVF_VMT * Prop_Rate_Auto / 1000000
'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"

'Safety Benefit Calculations
NPVF_Safety = NPVF_EnvBen
Fatality_Ben = VMT_I * Fat_Rate_Auto * a_rst("Fatality_Cost") * NPVF_Safety / 100000000
Injury_Ben = VMT_I * Inj_Rate_Auto * a_rst("Evident_Cost") * NPVF_Safety / 1000000
Prop_Ben = VMT_I * Prop_Rate_Auto * a_rst("PDO_Cost") * NPVF_Safety / 1000000
m_Safety_Ben = Fatality_Ben + Injury_Ben + Prop_Ben
'Set module level values (to be displayed)
m_Values.Add m_Safety_Ben, "SAFETY_BEN"
'Calculate Forecast Period
Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year") + 1

'Calculate economic analysis factors
PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / ((1 + Discount_Rate) ^ Forecast_Period))
PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Cost Calculations
'Capital Cost
For i = 1 To 5
    m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap + m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA

```

```

m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + m_Other1_Cost + m_Other2_Cost +
m_Other3_Cost

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
'Environmental Retrofit Costs

For i = 1 To 5
    m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * 1 / ((1 + Discount_Rate) ^
(2 * i - 1)))
    m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * 1 / ((1 + Discount_Rate)
^ (2 * i - 1)))
    m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * 1 / ((1 +
Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * a_rst("fishbarrier_bc")
m_StormWater_Ben = m_StormWater_Cap * a_rst("stormwater_bc")
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * a_rst("noisebarrier_bc")

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben + m_EnvRetrofit_Ben
m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"

```

```

m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

'First create all intermediate values used in the following calcs

'Calculate the Vital/Non-Vital Scores

If m_rst("Retire") = Yes Then

m_VitalScr = 100 - m_rst("CR_Vital")

m_NonVitalScr = 100 - m_rst("CR_NonVital")

Else

m_VitalScr = 0

m_NonVitalScr = 0

End If

'Calculation for the System Operation and Maintenance

m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)

m_Values.Add m_Sys_OM, "SYS_OM"

'Calculation for the System Preservation

m_Sys_Pres = m_rst("Q2A") * 50

m_Values.Add m_Sys_Pres, "SYS_PRES"

'Calculation for the Special Needs Transportation

m_Sp_Needs = 75 * m_rst("Q3A")

m_Values.Add m_Sp_Needs, "SP_NEEDS"

'Calculation for the Congestion Relief

If m_rst("WTP_Corridor") Then

If m_TT_Min > 0 Then

m_Cong_Rel = 100

Elseif m_TT_Min = 0 Then

m_Cong_Rel = 50

Else

m_Cong_Rel = 0

End If

Else

m_Cong_Rel = 0

End If

m_Values.Add m_Cong_Rel, "CONG_REL"

'Calculation for Increased Travel Options

If m_rst("Ferry_Class") = "Passenger Only Ferry" Then

m_Trav_Opt = 33

Else

m_Trav_Opt = 0

End If

m_Values.Add m_Trav_Opt, "TRAV_OPT"

'Calculation for Seamless Connections

m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)

m_Values.Add m_Seamless, "SEAMLESS"

'Calculation for Safety

If m_Safety_Ben > 0 Then


```

    SafetyA = 80
    Elseif m_Safety_Ben = 0 Then
        SafetyA = 40
    Else
        SafetyA = 0
    End If
    m_Safety = SafetyA + (20 * m_rst("Q7B"))
    m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
'The -1 Multiplication Corrects the negative sign introduced for true
    m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E")))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") > 0 Then
        m_Collab = 50
    Else
        m_Collab = 0
    End If
    m_Collab = m_Collab + (50 * m_rst("Q10A"))
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    St_Freight = m_rst("St_Freight ")
    If St_Freight < 6 And m_FrTT_Min > 0 Then
        FreightA = 45 + 45 * ((6 - St_Freight) / 5)
    Elseif St_Freight < 6 And m_FrTT_Min = 0 Then
        FreightA = 45
    Elseif St_Freight < 6 And m_FrTT_Min < 0 Then
        FreightA = 45 - 45 * ((6 - St_Freight) / 5)
    Else
        FreightA = 0
    End If
    m_Freight = FreightA + (10 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    Air_Quality = m_rst("Air_Quality")
    If m_Env_Ben > 0 Then
        If Air_Quality = 3 Then
            AirQualA = 80
        Elseif Air_Quality = 2 Then
            AirQualA = 60
        Else
            AirQualA = 40
        End If
    Elseif m_Env_Ben = 0 Then
        If Air_Quality = 3 Then
            AirQualA = 70
        Elseif Air_Quality = 2 Then
            AirQualA = 50
        Else
            AirQualA = 40
        End If
    End If

```

```

    End If
Else
    If Air_Quality = 3 Then
        AirQualA = 0
    ElseIf Air_Quality = 2 Then
        AirQualA = 20
    Else
        AirQualA = 40
    End If
End If
m_Air_Qual = AirQualA + (10 * m_rst("Q14A")) + (10 * m_rst("Q14B"))
m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
m_Wtr_Qual = (17 * m_rst("Q15A")) + (16 * m_rst("Q15B")) + _
    (16 * m_rst("Q15C"))
m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
HabitatA = ((5 * m_rst("Q16A")) + (5 * m_rst("Q16B")) + _
    (5 * m_rst("Q16C")) + (5 * m_rst("Q16D")))
If m_rst("Q16E") = 0 Then
    HabitatB = 40
ElseIf m_rst("Q16E") = 1 Then
    HabitatB = 20
ElseIf m_rst("Q16E") = 2 Then
    HabitatB = 10
Else
    HabitatB = 0
End If
If m_rst("Q16F") = 0 Then
    HabitatC = 40
ElseIf m_rst("Q16F") = 1 Then
    HabitatC = 20
ElseIf m_rst("Q16F") = 2 Then
    HabitatC = 10
Else
    HabitatC = 0
End If
m_Habitat = HabitatA + HabitatB + HabitatC
m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
m_Resource = 100 * m_rst("Q17")
m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub

```

Program Code for Highway Improvement Projects

Climbing Lane

Option Compare Database

'Variables for the class level database objects

```
Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection
```

'Variables used in intermediate calculations (variants)

```
Dim m_TT_Sav
Dim m_OpCost
Dim m_Approach
Dim m_WSDOT_Cap
Dim m_Federal_Cap
Dim m_Other1_Cap
Dim m_Other2_Cap
Dim m_Other3_Cap
Dim m_WSDOT_OM
Dim m_Federal_OM
Dim m_Other1_OM
Dim m_Other2_OM
Dim m_Other3_OM
Dim m_FishBarrier_Ben
Dim m_StormWater_Ben
Dim m_NoiseBarrier_Ben
Dim m_FishBarrier_Cap
Dim m_StormWater_Cap
Dim m_NoiseBarrier_Cap
```

'Variables to hold the calculated values (variants)

```
Dim m_Total_Benefit
Dim m_TT_Min
Dim m_TT_Ben
Dim m_User_Ben
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_EnvRetrofit_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
```

```
Dim m_Other2_Cost
Dim m_Other3_Cost
Dim m_Cap_Cost
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

'Variables to hold the outcome objective scores (variants)

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
Private Sub Class_Initialize()
    m_qryName = "hwy_calc_Climbing_Lane"
    m_calcsComplete = False
End Sub
```

Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef

    Set m_dbs = CurrentDb

    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid

    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then

        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection

        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
```

```

        atype = "prj_Global_Assumptions"
        pid = asmpnID
    End If

    'Open up a assumption recordset - depend on if in MICA or not
    Set qryDef = m_dbs.QueryDefs(atype)
    qryDef.Parameters!asmpnID = pid

    ' Set recordset to new values.
    Set a_rst = qryDef.OpenRecordset

    'Calculate all of the values for the project type
    CalculateBenefits
    CalculateOutObjScores

    'Check to see if the calcs are done for this project and set input status accordingly
    If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
    m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

    Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
    'Calculate Forecast Period
    Forecast_Period = 20

    'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / _
        ((1 + Discount_Rate) ^ Forecast_Period)))
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

    'Calculate traffic distribution - exit if no curve was selected
    'Travel Time Savings Calculations
    ADT_1 = m_rst("ADT_1")
    growth_rate = m_rst("growth_rate")

```

```

m_Length = m_rst("RoadLength")
AVO_auto = m_rst("avo_auto")
Truck_per = m_rst("Truck_per")
Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
Post_Speed_BC = m_rst("Post_Speed_BC")
Post_Speed_PC = m_rst("Post_Speed_PC")

'Determine operating cost values
Auto_OpCostHr_Full = a_rst("Veh_OpCostHr_Full")
Auto_OpCostHr_Direct = a_rst("Veh_OpCostHr_Direct")
Truck_OpCostHr_Full = a_rst("Truck_OpCostHr_Full")
Truck_OpCostHr_Direct = a_rst("Truck_OpCostHr_Direct")
  If a_rst("Full_Cost") Then
    m_Approach = "Full cost"
    OpCost_Auto = Auto_OpCostHr_Full
    OpCost_Truck = Truck_OpCostHr_Full
  Else
    m_Approach = "Direct cost"
    OpCost_Auto = Auto_OpCostHr_Direct
    OpCost_Truck = Truck_OpCostHr_Direct
  End If

'Calculate year 20 volume
ADT_20 = (ADT_1 * Forecast_Period * growth_rate) + ADT_1

'Calculate roadway capacities
Cap_NW_BC = m_rst("LnCap_NW_BC") * m_rst("NBWB_Lns_BC")
Cap_NW_PC = m_rst("LnCap_NW_PC") * m_rst("NBWB_Lns_PC")
Cap_SE_BC = m_rst("LnCap_SE_BC") * m_rst("SBEB_Lns_BC")
Cap_SE_PC = m_rst("LnCap_SE_PC") * m_rst("SBEB_Lns_PC")

Dim crvid As Integer

If Not IsNumeric(m_rst("Curve_ID")) Then Exit Sub
crvid = m_rst("Curve_ID")
'these are the arrays that hold all values to be used later
Dim Dir1 As Variant
Dim Dir2 As Variant

Dim NW_ivol_i(24) As Variant
Dim NW_ivol_f(24) As Variant
Dim SE_ivol_i(24) As Variant
Dim SE_ivol_f(24) As Variant

Dim NWvol_bci(24) As Variant
Dim NWvol_pci(24) As Variant
Dim NWvol_bcf(24) As Variant
Dim NWvol_pcf(24) As Variant
Dim SEvol_bci(24) As Variant
Dim SEvol_pci(24) As Variant
Dim SEvol_bcf(24) As Variant
Dim SEvol_pcf(24) As Variant

Dim NW_fvol_bci(24) As Variant
Dim NW_fvol_pci(24) As Variant
Dim NW_fvol_bcf(24) As Variant

```

```

Dim NW_fvol_pcf(24) As Variant
Dim SE_fvol_bci(24) As Variant
Dim SE_fvol_pci(24) As Variant
Dim SE_fvol_bcf(24) As Variant
Dim SE_fvol_pcf(24) As Variant

Dim GP_ttsav_i(24) As Variant
Dim GP_ttsav_f(24) As Variant
Dim Truck_ttsav_i(24) As Variant
Dim Truck_ttsav_f(24) As Variant

Dir1 = ReturnSpeedCurveDirectionData(crvid, s_NB_WB)
Dir2 = ReturnSpeedCurveDirectionData(crvid, s_SB_EB)

'calculate sum of traffic in both directions
For hr = 1 To 24
    Twoway_sum = Twoway_sum + Dir1(hr - 1) + Dir2(hr - 1)
Next hr

'calculate initial hourly volumes - not adjusted for congestion
For hr = 1 To 24
    NW_ivol_i(hr - 1) = (Dir1(hr - 1) / Twoway_sum) * ADT_1
    NW_ivol_f(hr - 1) = (Dir1(hr - 1) / Twoway_sum) * ADT_20
    SE_ivol_i(hr - 1) = (Dir2(hr - 1) / Twoway_sum) * ADT_1
    SE_ivol_f(hr - 1) = (Dir2(hr - 1) / Twoway_sum) * ADT_20
Next hr

'Redistribute volumes for hours in which v/c exceeds 1.2
'Initialize excess volume variables
excess_nw_bci = 0
excess_nw_pci = 0
excess_nw_bcf = 0
excess_nw_pcf = 0
excess_se_bci = 0
excess_se_pci = 0
excess_se_bcf = 0
excess_se_pcf = 0

'For hours 18 to 24, excess hourly volume shifts to following hour
For hr = 18 To 24
    NWvol_bci(hr - 1) = NW_ivol_i(hr - 1) + excess_nw_bci
    If NWvol_bci(hr - 1) > (Cap_NW_BC * 1.2) Then
        NW_fvol_bci(hr - 1) = (Cap_NW_BC * 1.2)
        excess_nw_bci = NWvol_bci(hr - 1) - (Cap_NW_BC * 1.2)
    Else
        NW_fvol_bci(hr - 1) = NWvol_bci(hr - 1)
        excess_nw_bci = 0
    End If

    NWvol_pci(hr - 1) = NW_ivol_i(hr - 1) + excess_nw_pci
    If NWvol_pci(hr - 1) > (Cap_NW_PC * 1.2) Then
        NW_fvol_pci(hr - 1) = (Cap_NW_PC * 1.2)
        excess_nw_pci = NWvol_pci(hr - 1) - (Cap_NW_PC * 1.2)
    Else
        NW_fvol_pci(hr - 1) = NWvol_pci(hr - 1)
        excess_nw_pci = 0
    End If

```

End If

```

NWvol_bcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nw_bcf
If NWvol_bcf(hr - 1) > (Cap_NW_BC * 1.2) Then
  NW_fvol_bcf(hr - 1) = (Cap_NW_BC * 1.2)
  excess_nw_bcf = NWvol_bcf(hr - 1) - (Cap_NW_BC * 1.2)
Else
  NW_fvol_bcf(hr - 1) = NWvol_bcf(hr - 1)
  excess_nw_bcf = 0
End If

```

```

NWvol_pcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nw_pcf
If NWvol_pcf(hr - 1) > (Cap_NW_PC * 1.2) Then
  NW_fvol_pcf(hr - 1) = (Cap_NW_PC * 1.2)
  excess_nw_pcf = NWvol_pcf(hr - 1) - (Cap_NW_PC * 1.2)
Else
  NW_fvol_pcf(hr - 1) = NWvol_pcf(hr - 1)
  excess_nw_pcf = 0
End If

```

```

SEvol_bci(hr - 1) = SE_ivol_i(hr - 1) + excess_se_bci
If SEvol_bci(hr - 1) > (Cap_SE_BC * 1.2) Then
  SE_fvol_bci(hr - 1) = (Cap_SE_BC * 1.2)
  excess_se_bci = SEvol_bci(hr - 1) - (Cap_SE_BC * 1.2)
Else
  SE_fvol_bci(hr - 1) = SEvol_bci(hr - 1)
  excess_se_bci = 0
End If

```

```

SEvol_pci(hr - 1) = SE_ivol_i(hr - 1) + excess_se_pci
If SEvol_pci(hr - 1) > (Cap_SE_PC * 1.2) Then
  SE_fvol_pci(hr - 1) = (Cap_SE_PC * 1.2)
  excess_se_pci = SEvol_pci(hr - 1) - (Cap_SE_PC * 1.2)
Else
  SE_fvol_pci(hr - 1) = SEvol_pci(hr - 1)
  excess_se_pci = 0
End If

```

```

SEvol_bcf(hr - 1) = SE_ivol_f(hr - 1) + excess_se_bcf
If SEvol_bcf(hr - 1) > (Cap_SE_BC * 1.2) Then
  SE_fvol_bcf(hr - 1) = (Cap_SE_BC * 1.2)
  excess_se_bcf = SEvol_bcf(hr - 1) - (Cap_SE_BC * 1.2)
Else
  SE_fvol_bcf(hr - 1) = SEvol_bcf(hr - 1)
  excess_se_bcf = 0
End If

```

```

SEvol_pcf(hr - 1) = SE_ivol_f(hr - 1) + excess_se_pcf
If SEvol_pcf(hr - 1) > (Cap_SE_PC * 1.2) Then
  SE_fvol_pcf(hr - 1) = (Cap_SE_PC * 1.2)
  excess_se_pcf = NWvol_pcf(hr - 1) - (Cap_NW_PC * 1.2)
Else
  SE_fvol_pcf(hr - 1) = SEvol_pcf(hr - 1)
  excess_se_pcf = 0
End If

```


Next hr

'For hours 1 to 17, excess hourly volume shifts to preceeding hour

'Reset excess volume variables

```

excess_nw_bci = 0
excess_nw_pci = 0
excess_nw_bcf = 0
excess_nw_pcf = 0
excess_se_bci = 0
excess_se_pci = 0
excess_se_bcf = 0
excess_se_pcf = 0

```

For hr = 17 To 1 Step -1

```

NWvol_bci(hr - 1) = NW_ivol_i(hr - 1) + excess_nw_bci
If NWvol_bci(hr - 1) > (Cap_NW_BC * 1.2) Then
  NW_fvol_bci(hr - 1) = (Cap_NW_BC * 1.2)
  excess_nw_bci = NWvol_bci(hr - 1) - (Cap_NW_BC * 1.2)
Else
  NW_fvol_bci(hr - 1) = NWvol_bci(hr - 1)
  excess_nw_bci = 0
End If

```

```

NWvol_pci(hr - 1) = NW_ivol_i(hr - 1) + excess_nw_pci
If NWvol_pci(hr - 1) > (Cap_NW_PC * 1.2) Then
  NW_fvol_pci(hr - 1) = (Cap_NW_PC * 1.2)
  excess_nw_pci = NWvol_pci(hr - 1) - (Cap_NW_PC * 1.2)
Else
  NW_fvol_pci(hr - 1) = NWvol_pci(hr - 1)
  excess_nw_pci = 0
End If

```

```

NWvol_bcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nw_bcf
If NWvol_bcf(hr - 1) > (Cap_NW_BC * 1.2) Then
  NW_fvol_bcf(hr - 1) = (Cap_NW_BC * 1.2)
  excess_nw_bcf = NWvol_bcf(hr - 1) - (Cap_NW_BC * 1.2)
Else
  NW_fvol_bcf(hr - 1) = NWvol_bcf(hr - 1)
  excess_nw_bcf = 0
End If

```

```

NWvol_pcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nw_pcf
If NWvol_pcf(hr - 1) > (Cap_NW_PC * 1.2) Then
  NW_fvol_pcf(hr - 1) = (Cap_NW_PC * 1.2)
  excess_nw_pcf = NWvol_pcf(hr - 1) - (Cap_NW_PC * 1.2)
Else
  NW_fvol_pcf(hr - 1) = NWvol_pcf(hr - 1)
  excess_nw_pcf = 0
End If

```

```

SEvol_bci(hr - 1) = SE_ivol_i(hr - 1) + excess_se_bci
If SEvol_bci(hr - 1) > (Cap_SE_BC * 1.2) Then
  SE_fvol_bci(hr - 1) = (Cap_SE_BC * 1.2)
  excess_se_bci = SEvol_bci(hr - 1) - (Cap_SE_BC * 1.2)
Else
  SE_fvol_bci(hr - 1) = SEvol_bci(hr - 1)

```

```

    excess_se_bci = 0
End If

SEvol_pci(hr - 1) = SE_ivol_i(hr - 1) + excess_se_pci
If SEvol_pci(hr - 1) > (Cap_SE_PC * 1.2) Then
    SE_fvol_pci(hr - 1) = (Cap_SE_PC * 1.2)
    excess_se_pci = SEvol_pci(hr - 1) - (Cap_SE_PC * 1.2)
Else
    SE_fvol_pci(hr - 1) = SEvol_pci(hr - 1)
    excess_se_pci = 0
End If

SEvol_bcf(hr - 1) = SE_ivol_f(hr - 1) + excess_se_bcf
If SEvol_bcf(hr - 1) > (Cap_SE_BC * 1.2) Then
    SE_fvol_bcf(hr - 1) = (Cap_SE_BC * 1.2)
    excess_se_bcf = SEvol_bcf(hr - 1) - (Cap_SE_BC * 1.2)
Else
    SE_fvol_bcf(hr - 1) = SEvol_bcf(hr - 1)
    excess_se_bcf = 0
End If

SEvol_pcf(hr - 1) = SE_ivol_f(hr - 1) + excess_se_pcf
If SEvol_pcf(hr - 1) > (Cap_SE_PC * 1.2) Then
    SE_fvol_pcf(hr - 1) = (Cap_SE_PC * 1.2)
    excess_se_pcf = SEvol_pcf(hr - 1) - (Cap_SE_PC * 1.2)
Else
    SE_fvol_pcf(hr - 1) = SEvol_pcf(hr - 1)
    excess_se_pcf = 0
End If

Next hr

'calculate factors that are used to calculate travel times
' these pieces are same in all calcs
cartruck_tt_factor = m_Length * (1 - Truck_per) * Truck_per * (1 - Truck_per) * _
(Post_Speed_PC / Post_Speed_BC)
caronly_tt_factor = m_Length * (1 - Truck_per) * (1 - (Truck_per * (1 - Truck_per) * _
(Post_Speed_PC / Post_Speed_BC)))
truckonly_tt_factor = m_Length * Truck_per

'initiate the speed curve class
If IsNull(m_rst("facility_BC")) Or IsNull(m_rst("facility_PC")) Then Exit Sub

Dim spCurveBC As New SpeedCurve
Dim spCurvePC As New SpeedCurve
spCurveBC.SetHighwayClass (m_rst("facility_BC"))
spCurvePC.SetHighwayClass (m_rst("facility_PC"))

'calculate vc ratios, operating speed (from curves), and travel time savings
For hr = 1 To 24

    NW_vc_bci = NW_fvol_bci(hr - 1) / Cap_NW_BC
    NW_vc_pci = NW_fvol_pci(hr - 1) / Cap_NW_PC
    NW_vc_bcf = NW_fvol_bcf(hr - 1) / Cap_NW_BC
    NW_vc_pcf = NW_fvol_pcf(hr - 1) / Cap_NW_PC

```

```

SE_vc_bci = SE_fvol_bci(hr - 1) / Cap_SE_BC
SE_vc_pci = SE_fvol_pci(hr - 1) / Cap_SE_PC
SE_vc_bcf = SE_fvol_bcf(hr - 1) / Cap_SE_BC
SE_vc_pcf = SE_fvol_pcf(hr - 1) / Cap_SE_PC

```

```

NW_os_bci = spCurveBC.RetrieveSpeedData(NW_vc_bci)
NW_os_pci = spCurvePC.RetrieveSpeedData(NW_vc_pci)
NW_os_bcf = spCurveBC.RetrieveSpeedData(NW_vc_bcf)
NW_os_pcf = spCurvePC.RetrieveSpeedData(NW_vc_pcf)
SE_os_bci = spCurveBC.RetrieveSpeedData(SE_vc_bci)
SE_os_pci = spCurvePC.RetrieveSpeedData(SE_vc_pci)
SE_os_bcf = spCurveBC.RetrieveSpeedData(SE_vc_bcf)
SE_os_pcf = spCurvePC.RetrieveSpeedData(SE_vc_pcf)

```

```

GP_ttsav_i(hr - 1) = (((NW_fvol_bci(hr - 1) / NW_os_bci) -
(NW_fvol_pci(hr - 1) / NW_os_pci)) +
((SE_fvol_bci(hr - 1) / SE_os_bci) -
(SE_fvol_pci(hr - 1) / SE_os_pci))) *
(cartruck_tt_factor + caronly_tt_factor)

```

```

GP_ttsav_f(hr - 1) = (((NW_fvol_bcf(hr - 1) / NW_os_bcf) -
(NW_fvol_pcf(hr - 1) / NW_os_pcf)) +
((SE_fvol_bcf(hr - 1) / SE_os_bcf) -
(SE_fvol_pcf(hr - 1) / SE_os_pcf))) *
(cartruck_tt_factor + caronly_tt_factor)

```

```

Truck_ttsav_i(hr - 1) = (((NW_fvol_bci(hr - 1) / NW_os_bci) -
(NW_fvol_pci(hr - 1) / NW_os_pci)) +
((SE_fvol_bci(hr - 1) / SE_os_bci) -
(SE_fvol_pci(hr - 1) / SE_os_pci))) *
truckonly_tt_factor

```

```

Truck_ttsav_f(hr - 1) = (((NW_fvol_bcf(hr - 1) / NW_os_bcf) -
(NW_fvol_pcf(hr - 1) / NW_os_pcf)) +
((SE_fvol_bcf(hr - 1) / SE_os_bcf) -
(SE_fvol_pcf(hr - 1) / SE_os_pcf))) *
truckonly_tt_factor

```

'Total travel time savings - initial and final year

```

ttsav_auto_i = ttsav_auto_i + (GP_ttsav_i(hr - 1) * Ann_Daily_Benefit)
ttsav_auto_f = ttsav_auto_f + (GP_ttsav_f(hr - 1) * Ann_Daily_Benefit)
ttsav_truck_i = ttsav_truck_i + (Truck_ttsav_i(hr - 1) * Ann_Daily_Benefit)
ttsav_truck_f = ttsav_truck_f + (Truck_ttsav_f(hr - 1) * Ann_Daily_Benefit)

```

Next hr

```

Set spCurveBC = Nothing
Set spCurvePC = Nothing

```

'Total travel time benefit - initial and final year

```

Time_Value_Truck = a_rst("Time_Value_Truck")
Time_Value_Pers = a_rst("Time_Value_Pers")
tt_ben_i = (ttsav_truck_i * Time_Value_Truck) +
(ttsav_auto_i * AVO_auto * Time_Value_Pers)
tt_ben_f = (ttsav_truck_f * Time_Value_Truck) +

```

```

      (ttsav_auto_f * AVO_auto * Time_Value_Pers)

'Total Travel Time Calculations
  m_TT_Sav = ((ttsav_auto_i + ttsav_truck_i) + _
    (ttsav_auto_f + ttsav_truck_f)) * Forecast_Period / 2
  m_TT_Min = m_TT_Sav * 60

  G_tt_ben = (tt_ben_f - tt_ben_i) / (Forecast_Period - 1)
  m_TT_Ben = (tt_ben_i * PofA) + (G_tt_ben * PofG)

'Total operating cost savings - initial and final year

  op_cost_i = (ttsav_truck_i * OpCost_Truck) + _
    (ttsav_auto_i * OpCost_Auto)
  op_cost_f = (ttsav_truck_f * OpCost_Truck) + _
    (ttsav_auto_f * OpCost_Auto)

'User Benefit Calculations
  G_Opcost_Ben = (op_cost_f - op_cost_i) / (Forecast_Period - 1)
  m_User_Ben = (op_cost_i * PofA) + (G_Opcost_Ben * PofG)

'Set module level values (to be displayed)
  m_Values.Add m_TT_Sav, "TT_SAV"
  m_Values.Add m_TT_Min, "TT_MIN"
  m_Values.Add m_TT_Ben, "TT_BEN"
  m_Values.Add m_User_Ben, "USER_BEN"

'Air Pollution Calculations - for diverted auto trips -
'none in current procedure, but keep as placeholder
  m_CO_Tons = 0
  m_VOC_Tons = 0
  m_NOX_Tons = 0
  m_PM10_Tons = 0
  m_Env_Ben = 0

'Set module level values (to be displayed)
  m_Values.Add m_CO_Tons, "CO_TONS"
  m_Values.Add m_VOC_Tons, "VOC_TONS"
  m_Values.Add m_NOX_Tons, "NOX_TONS"
  m_Values.Add m_PM10_Tons, "PM10_TONS"
  m_Values.Add m_Env_Ben, "ENV_BEN"

'Safety Calculations
  Fatality_Cost = a_rst("Fatality_Cost")
  Disable_Cost = a_rst("Disable_Cost")
  Evident_Cost = a_rst("Evident_Cost")
  Possible_Cost = a_rst("Possible_Cost")
  PDO_Cost = a_rst("PDO_Cost")

'Accident reduction factors
  R = 0
  RP = 0

  Dim whereSQL As String
  whereSQL = ReturnSafetyWhereSQL(m_rst)
  If whereSQL <> "" Then

```

```

'Get 1-d array with values or zero for 5 improvement types
Rarr = ReturnSafetyArray(whereSQL, "Fat_Inj_Red")
RParr = ReturnSafetyArray(whereSQL, "PDO_Red")

If IsArray(Rarr) Then
  For i = 0 To 4
    If i <> 0 Then
      'For fatality and injury accidents
      Rterm = Rterm * ((100 - Rarr(i - 1)) / 100)
      R = R + (Rarr(i) * Rterm)
      'For property damage accidents
      RPterm = RPterm * ((100 - RParr(i - 1)) / 100)
      RP = RP + (RParr(i) * Rterm)
    Else
      'For fatality and injury accidents
      R = Rarr(i)
      Rterm = 1
      'For property damage accidents
      RP = RParr(i)
      RPterm = 1
    End If
  Next
End If

'Accident reduction due to proposed project
ann_fatality = m_rst("3yr_Fatality") * (R / 100) / 3
ann_disable = m_rst("3yr_Disable") * (R / 100) / 3
ann_evident = m_rst("3yr_Evident") * (R / 100) / 3
ann_possible = m_rst("3yr_Possible") * (R / 100) / 3
ann_property = m_rst("3yr_Property") * (RP / 100) / 3

m_Fatality = (-1) * ann_fatality * 20
m_Injury = (-1) * (ann_disable + ann_evident + ann_possible) * 20
m_Property = (-1) * ann_property * 20

'Calculate safety benefits
m_Safety_Ben = ((ann_fatality * Fatality_Cost) + _
(ann_disable * Disable_Cost) + (ann_evident * Evident_Cost) + _
(ann_possible * Possible_Cost) + (ann_property * PDO_Cost)) * PofA

'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
'Capital Cost
For i = 1 To 5
  m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
  1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
  1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
  1 / ((1 + Discount_Rate) ^ (2 * i - 1)))

```

```

    m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
      1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
      1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  Next
  m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
  m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

  Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
  m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations

```

```

    Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
    If (m_Total_Cost - m_Terminal_Cost) = 0 Then
        m_BCR = 0
    Else
        m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
    End If
    If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
        m_WSDOT_BCR = 0
    Else
        m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
    End If
    'Set module level values (to be displayed)
    m_Values.Add m_Total_Cost, "TOTAL_COST"
    m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
    m_Values.Add m_Federal_Cost, "FEDERAL_COST"
    m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
    m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
    m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
    m_Values.Add m_Other1_Cost, "OTHER1_COST"
    m_Values.Add m_Other2_Cost, "OTHER2_COST"
    m_Values.Add m_Other3_Cost, "OTHER3_COST"
    m_Values.Add m_Cap_Cost, "CAP_COST"
    m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
    m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
    m_Values.Add m_Other1_Cap, "OTHER1_CAP"
    m_Values.Add m_Other2_Cap, "OTHER2_CAP"
    m_Values.Add m_Other3_Cap, "OTHER3_CAP"
    m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
    m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
    m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
    m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
    m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
    m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
    m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
    m_Values.Add m_WSDOT_OM, "WSDOT_OM"
    m_Values.Add m_Federal_OM, "FEDERAL_OM"
    m_Values.Add m_Other1_OM, "OTHER1_OM"
    m_Values.Add m_Other2_OM, "OTHER2_OM"
    m_Values.Add m_Other3_OM, "OTHER3_OM"
    m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
    m_Values.Add m_BCR, "BCR"
    m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

```
Private Sub CalculateOutObjScores()
```

```
On Error Resume Next
```

```
'Calculation for the System Operation and Maintenance
```

```
m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
```

```
m_Values.Add m_Sys_OM, "SYS_OM"
```

```
'Calculation for the System Preservation
```

```
m_Sys_Pres = 100 * m_rst("Q2A")
```

```
m_Values.Add m_Sys_Pres, "SYS_PRES"
```

```
'Calculation for the Special Needs Transportation
```

```
m_Sp_Needs = 100 * m_rst("Q3A")
```

```

    m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
  Else
    m_Cong_Rel = (m_rst("Q4") * 50)
  End If
  m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
  Else
    m_Safety = m_rst("Q7B") * 50
  End If
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  'The -1 Multiplication Corrects the negative sign introduced for true
  m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
  Else
    m_Collab = (m_rst("Q10A") * 50) + 50
  End If
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
  If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + m_rst("Q14A") * 50
  Else
    m_Air_Qual = 0
  End If
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
    (33 * m_rst("Q15C"))

```



```
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"  
'Calculation for the Habitat  
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _  
                (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _  
                (1 + m_rst("Q16E") + m_rst("Q16F"))  
    m_Values.Add m_Habitat, "HABITAT"  
'Calculation for the Use of Resources  
    m_Resource = 100 * m_rst("Q17")  
    m_Values.Add m_Resource, "RESOURCE"  
  
End Sub  
  
Public Function GetItemValue(itemname As String) As Variant  
    Dim rtnval  
  
    rtnval = m_Values.Retrieve(itemname)  
  
    GetItemValue = rtnval  
End Function  
  
Private Sub Class_Terminate()  
    a_rst.Close  
    m_rst.Close  
    m_dbs.Close  
    Set a_rst = Nothing  
    Set m_rst = Nothing  
    Set m_dbs = Nothing  
    Set m_Values = Nothing  
End Sub
```

General Purpose Lane

Option Compare Database

'Variables for the class level database objects

```
Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection
```

'Variables used in intermediate calculations (variants)

```
Dim m_TT_Sav
Dim m_OpCost
Dim m_Approach
Dim m_WSDOT_Cap
Dim m_Federal_Cap
Dim m_Other1_Cap
Dim m_Other2_Cap
Dim m_Other3_Cap
Dim m_WSDOT_OM
Dim m_Federal_OM
Dim m_Other1_OM
Dim m_Other2_OM
Dim m_Other3_OM
Dim m_FishBarrier_Ben
Dim m_StormWater_Ben
Dim m_NoiseBarrier_Ben
Dim m_FishBarrier_Cap
Dim m_StormWater_Cap
Dim m_NoiseBarrier_Cap
```

'Variables to hold the calculated values (variants)

```
Dim m_Total_Benefit
Dim m_TT_Min
Dim m_TT_Ben
Dim m_User_Ben
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_EnvRetrofit_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
Dim m_Other2_Cost
Dim m_Other3_Cost
```

```
Dim m_Cap_Cost
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
Private Sub Class_Initialize()
    m_qryName = "hwy_calc_GP_Lane"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If
```

```

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
    Forecast_Period = 20

'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / _
        ((1 + Discount_Rate) ^ Forecast_Period)))
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Calculate traffic distribution - exit if no curve was selected
'Travel Time Savings Calculations
    ADT_1 = m_rst("ADT_1")
    growth_rate = m_rst("growth_rate")
    m_Length = m_rst("RoadLength")
    AVO_auto = m_rst("avo_auto")

```

```

'Determine operating cost values
Auto_OpCostHr_Full = a_rst("Veh_OpCostHr_Full")
Auto_OpCostHr_Direct = a_rst("Veh_OpCostHr_Direct")
Truck_OpCostHr_Full = a_rst("Truck_OpCostHr_Full")
Truck_OpCostHr_Direct = a_rst("Truck_OpCostHr_Direct")
  If a_rst("Full_Cost") Then
    m_Approach = "Full cost"
    OpCost_Auto = Auto_OpCostHr_Full
    OpCost_Truck = Truck_OpCostHr_Full
  Else
    m_Approach = "Direct cost"
    OpCost_Auto = Auto_OpCostHr_Direct
    OpCost_Truck = Truck_OpCostHr_Direct
  End If

'Calculate year 20 volume
ADT_20 = (ADT_1 * Forecast_Period * growth_rate) + ADT_1

'Calculate roadway capacities
Cap_NW_BC = m_rst("LnCap_NW_BC") * m_rst("NBWB_Lns_BC")
Cap_NW_PC = m_rst("LnCap_NW_PC") * m_rst("NBWB_Lns_PC")
Cap_SE_BC = m_rst("LnCap_SE_BC") * m_rst("SBEB_Lns_BC")
Cap_SE_PC = m_rst("LnCap_SE_PC") * m_rst("SBEB_Lns_PC")

Dim crvid As Integer

If Not IsNumeric(m_rst("Curve_ID")) Then Exit Sub
crvid = m_rst("Curve_ID")
'these are the arrays that hold all values to be used later
Dim Dir1 As Variant
Dim Dir2 As Variant

Dim NW_ivol_i(24) As Variant
Dim NW_ivol_f(24) As Variant
Dim SE_ivol_i(24) As Variant
Dim SE_ivol_f(24) As Variant

Dim NWvol_bci(24) As Variant
Dim NWvol_pci(24) As Variant
Dim NWvol_bcf(24) As Variant
Dim NWvol_pcf(24) As Variant
Dim SEvol_bci(24) As Variant
Dim SEvol_pci(24) As Variant
Dim SEvol_bcf(24) As Variant
Dim SEvol_pcf(24) As Variant

Dim NW_fvol_bci(24) As Variant
Dim NW_fvol_pci(24) As Variant
Dim NW_fvol_bcf(24) As Variant
Dim NW_fvol_pcf(24) As Variant
Dim SE_fvol_bci(24) As Variant
Dim SE_fvol_pci(24) As Variant
Dim SE_fvol_bcf(24) As Variant
Dim SE_fvol_pcf(24) As Variant

Dim NW_ttsav_i(24) As Variant

```

```

Dim NW_ttsav_f(24) As Variant
Dim SE_ttsav_i(24) As Variant
Dim SE_ttsav_f(24) As Variant

Dir1 = ReturnSpeedCurveDirectionData(crvid, s_NB_WB)
Dir2 = ReturnSpeedCurveDirectionData(crvid, s_SB_EB)

'calculate sum of traffic in both directions
For hr = 1 To 24
    Twoway_sum = Twoway_sum + Dir1(hr - 1) + Dir2(hr - 1)
Next hr

'calculate initial hourly volumes - not adjusted for congestion
For hr = 1 To 24
    NW_ivol_i(hr - 1) = (Dir1(hr - 1) / Twoway_sum) * ADT_1
    NW_ivol_f(hr - 1) = (Dir1(hr - 1) / Twoway_sum) * ADT_20
    SE_ivol_i(hr - 1) = (Dir2(hr - 1) / Twoway_sum) * ADT_1
    SE_ivol_f(hr - 1) = (Dir2(hr - 1) / Twoway_sum) * ADT_20
Next hr

'Redistribute volumes for hours in which v/c exceeds 1.2
'Initialize excess volume variables
excess_nw_bci = 0
excess_nw_pci = 0
excess_nw_bcf = 0
excess_nw_pcf = 0
excess_se_bci = 0
excess_se_pci = 0
excess_se_bcf = 0
excess_se_pcf = 0

'For hours 18 to 24, excess hourly volume shifts to following hour
For hr = 18 To 24
    NWvol_bci(hr - 1) = NW_ivol_i(hr - 1) + excess_nw_bci
    If NWvol_bci(hr - 1) > (Cap_NW_BC * 1.2) Then
        NW_fvol_bci(hr - 1) = (Cap_NW_BC * 1.2)
        excess_nw_bci = NWvol_bci(hr - 1) - (Cap_NW_BC * 1.2)
    Else
        NW_fvol_bci(hr - 1) = NWvol_bci(hr - 1)
        excess_nw_bci = 0
    End If

    NWvol_pci(hr - 1) = NW_ivol_i(hr - 1) + excess_nw_pci
    If NWvol_pci(hr - 1) > (Cap_NW_PC * 1.2) Then
        NW_fvol_pci(hr - 1) = (Cap_NW_PC * 1.2)
        excess_nw_pci = NWvol_pci(hr - 1) - (Cap_NW_PC * 1.2)
    Else
        NW_fvol_pci(hr - 1) = NWvol_pci(hr - 1)
        excess_nw_pci = 0
    End If

    NWvol_bcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nw_bcf
    If NWvol_bcf(hr - 1) > (Cap_NW_BC * 1.2) Then
        NW_fvol_bcf(hr - 1) = (Cap_NW_BC * 1.2)
        excess_nw_bcf = NWvol_bcf(hr - 1) - (Cap_NW_BC * 1.2)
    Else

```

```

NW_fvol_bcf(hr - 1) = NWvol_bcf(hr - 1)
excess_nw_bcf = 0
End If

NWvol_pcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nw_pcf
If NWvol_pcf(hr - 1) > (Cap_NW_PC * 1.2) Then
  NW_fvol_pcf(hr - 1) = (Cap_NW_PC * 1.2)
  excess_nw_pcf = NWvol_pcf(hr - 1) - (Cap_NW_PC * 1.2)
Else
  NW_fvol_pcf(hr - 1) = NWvol_pcf(hr - 1)
  excess_nw_pcf = 0
End If

```

```

SEvol_bci(hr - 1) = SE_ivol_i(hr - 1) + excess_se_bci
If SEvol_bci(hr - 1) > (Cap_SE_BC * 1.2) Then
  SE_fvol_bci(hr - 1) = (Cap_SE_BC * 1.2)
  excess_se_bci = SEvol_bci(hr - 1) - (Cap_SE_BC * 1.2)
Else
  SE_fvol_bci(hr - 1) = SEvol_bci(hr - 1)
  excess_se_bci = 0
End If

```

```

SEvol_pci(hr - 1) = SE_ivol_i(hr - 1) + excess_se_pci
If SEvol_pci(hr - 1) > (Cap_SE_PC * 1.2) Then
  SE_fvol_pci(hr - 1) = (Cap_SE_PC * 1.2)
  excess_se_pci = SEvol_pci(hr - 1) - (Cap_SE_PC * 1.2)
Else
  SE_fvol_pci(hr - 1) = SEvol_pci(hr - 1)
  excess_se_pci = 0
End If

```

```

SEvol_bcf(hr - 1) = SE_ivol_f(hr - 1) + excess_se_bcf
If SEvol_bcf(hr - 1) > (Cap_SE_BC * 1.2) Then
  SE_fvol_bcf(hr - 1) = (Cap_SE_BC * 1.2)
  excess_se_bcf = SEvol_bcf(hr - 1) - (Cap_SE_BC * 1.2)
Else
  SE_fvol_bcf(hr - 1) = SEvol_bcf(hr - 1)
  excess_se_bcf = 0
End If

```

```

SEvol_pcf(hr - 1) = SE_ivol_f(hr - 1) + excess_se_pcf
If SEvol_pcf(hr - 1) > (Cap_SE_PC * 1.2) Then
  SE_fvol_pcf(hr - 1) = (Cap_SE_PC * 1.2)
  excess_se_pcf = NWvol_pcf(hr - 1) - (Cap_NW_PC * 1.2)
Else
  SE_fvol_pcf(hr - 1) = SEvol_pcf(hr - 1)
  excess_se_pcf = 0
End If

```

Next hr

'For hours 1 to 17, excess hourly volume shifts to preceeding hour

'Reset excess volume variables

```

excess_nw_bci = 0
excess_nw_pci = 0
excess_nw_bcf = 0

```

```

excess_nw_pcf = 0
excess_se_bci = 0
excess_se_pci = 0
excess_se_bcf = 0
excess_se_pcf = 0

```

```

For hr = 17 To 1 Step -1

```

```

  NWvol_bci(hr - 1) = NW_ivol_i(hr - 1) + excess_nw_bci
  If NWvol_bci(hr - 1) > (Cap_NW_BC * 1.2) Then
    NW_fvol_bci(hr - 1) = (Cap_NW_BC * 1.2)
    excess_nw_bci = NWvol_bci(hr - 1) - (Cap_NW_BC * 1.2)
  Else
    NW_fvol_bci(hr - 1) = NWvol_bci(hr - 1)
    excess_nw_bci = 0
  End If

```

```

  NWvol_pci(hr - 1) = NW_ivol_i(hr - 1) + excess_nw_pci
  If NWvol_pci(hr - 1) > (Cap_NW_PC * 1.2) Then
    NW_fvol_pci(hr - 1) = (Cap_NW_PC * 1.2)
    excess_nw_pci = NWvol_pci(hr - 1) - (Cap_NW_PC * 1.2)
  Else
    NW_fvol_pci(hr - 1) = NWvol_pci(hr - 1)
    excess_nw_pci = 0
  End If

```

```

  NWvol_bcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nw_bcf
  If NWvol_bcf(hr - 1) > (Cap_NW_BC * 1.2) Then
    NW_fvol_bcf(hr - 1) = (Cap_NW_BC * 1.2)
    excess_nw_bcf = NWvol_bcf(hr - 1) - (Cap_NW_BC * 1.2)
  Else
    NW_fvol_bcf(hr - 1) = NWvol_bcf(hr - 1)
    excess_nw_bcf = 0
  End If

```

```

  NWvol_pcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nw_pcf
  If NWvol_pcf(hr - 1) > (Cap_NW_PC * 1.2) Then
    NW_fvol_pcf(hr - 1) = (Cap_NW_PC * 1.2)
    excess_nw_pcf = NWvol_pcf(hr - 1) - (Cap_NW_PC * 1.2)
  Else
    NW_fvol_pcf(hr - 1) = NWvol_pcf(hr - 1)
    excess_nw_pcf = 0
  End If

```

```

  SEvol_bci(hr - 1) = SE_ivol_i(hr - 1) + excess_se_bci
  If SEvol_bci(hr - 1) > (Cap_SE_BC * 1.2) Then
    SE_fvol_bci(hr - 1) = (Cap_SE_BC * 1.2)
    excess_se_bci = SEvol_bci(hr - 1) - (Cap_SE_BC * 1.2)
  Else
    SE_fvol_bci(hr - 1) = SEvol_bci(hr - 1)
    excess_se_bci = 0
  End If

```

```

  SEvol_pci(hr - 1) = SE_ivol_i(hr - 1) + excess_se_pci
  If SEvol_pci(hr - 1) > (Cap_SE_PC * 1.2) Then
    SE_fvol_pci(hr - 1) = (Cap_SE_PC * 1.2)
    excess_se_pci = SEvol_pci(hr - 1) - (Cap_SE_PC * 1.2)

```



```

Else
  SE_fvol_pci(hr - 1) = SEvol_pci(hr - 1)
  excess_se_pci = 0
End If

SEvol_bcf(hr - 1) = SE_ivol_f(hr - 1) + excess_se_bcf
If SEvol_bcf(hr - 1) > (Cap_SE_BC * 1.2) Then
  SE_fvol_bcf(hr - 1) = (Cap_SE_BC * 1.2)
  excess_se_bcf = SEvol_bcf(hr - 1) - (Cap_SE_BC * 1.2)
Else
  SE_fvol_bcf(hr - 1) = SEvol_bcf(hr - 1)
  excess_se_bcf = 0
End If

SEvol_pcf(hr - 1) = SE_ivol_f(hr - 1) + excess_se_pcf
If SEvol_pcf(hr - 1) > (Cap_SE_PC * 1.2) Then
  SE_fvol_pcf(hr - 1) = (Cap_SE_PC * 1.2)
  excess_se_pcf = SEvol_pcf(hr - 1) - (Cap_SE_PC * 1.2)
Else
  SE_fvol_pcf(hr - 1) = SEvol_pcf(hr - 1)
  excess_se_pcf = 0
End If

Next hr

'initiate the speed curve class
If IsNull(m_rst("facility_BC")) Or IsNull(m_rst("facility_PC")) Then Exit Sub

Dim spCurveBC As New SpeedCurve
Dim spCurvePC As New SpeedCurve
spCurveBC.SetHighwayClass (m_rst("facility_BC"))
spCurvePC.SetHighwayClass (m_rst("facility_PC"))

'calculate vc ratios, operating speed (from curves), and travel time savings
For hr = 1 To 24

  NW_vc_bci = NW_fvol_bci(hr - 1) / Cap_NW_BC
  NW_vc_pci = NW_fvol_pci(hr - 1) / Cap_NW_PC
  NW_vc_bcf = NW_fvol_bcf(hr - 1) / Cap_NW_BC
  NW_vc_pcf = NW_fvol_pcf(hr - 1) / Cap_NW_PC

  SE_vc_bci = SE_fvol_bci(hr - 1) / Cap_SE_BC
  SE_vc_pci = SE_fvol_pci(hr - 1) / Cap_SE_PC
  SE_vc_bcf = SE_fvol_bcf(hr - 1) / Cap_SE_BC
  SE_vc_pcf = SE_fvol_pcf(hr - 1) / Cap_SE_PC

  NW_os_bci = spCurveBC.RetrieveSpeedData(NW_vc_bci)
  NW_os_pci = spCurvePC.RetrieveSpeedData(NW_vc_pci)
  NW_os_bcf = spCurveBC.RetrieveSpeedData(NW_vc_bcf)
  NW_os_pcf = spCurvePC.RetrieveSpeedData(NW_vc_pcf)
  SE_os_bci = spCurveBC.RetrieveSpeedData(SE_vc_bci)
  SE_os_pci = spCurvePC.RetrieveSpeedData(SE_vc_pci)
  SE_os_bcf = spCurveBC.RetrieveSpeedData(SE_vc_bcf)
  SE_os_pcf = spCurvePC.RetrieveSpeedData(SE_vc_pcf)

  NW_ttsav_i(hr - 1) = ((NW_fvol_bci(hr - 1) / NW_os_bci) - _

```

```

(NW_fvol_pci(hr - 1) / NW_os_pci)) * m_Length
NW_ttsav_f(hr - 1) = ((NW_fvol_bcf(hr - 1) / NW_os_bcf) - _
(NW_fvol_pcf(hr - 1) / NW_os_pcf)) * m_Length
SE_ttsav_i(hr - 1) = ((SE_fvol_bci(hr - 1) / SE_os_bci) - _
(SE_fvol_pci(hr - 1) / SE_os_pci)) * m_Length
SE_ttsav_f(hr - 1) = ((SE_fvol_bcf(hr - 1) / SE_os_bcf) - _
(SE_fvol_pcf(hr - 1) / SE_os_pcf)) * m_Length

```

'Total travel time savings - initial and final year

```

tt_sav_i = tt_sav_i + NW_ttsav_i(hr - 1) + SE_ttsav_i(hr - 1)
tt_sav_f = tt_sav_f + NW_ttsav_f(hr - 1) + SE_ttsav_f(hr - 1)

```

Next hr

Set spCurveBC = Nothing

Set spCurvePC = Nothing

'Allot travel time savings between truck and auto

```

Truck_per = m_rst("Truck_per")
Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
ttsav_truck_i = Truck_per * tt_sav_i * Ann_Daily_Benefit
ttsav_auto_i = (1 - Truck_per) * tt_sav_i * Ann_Daily_Benefit
ttsav_truck_f = Truck_per * tt_sav_f * Ann_Daily_Benefit
ttsav_auto_f = (1 - Truck_per) * tt_sav_f * Ann_Daily_Benefit

```

'Total travel time benefit - initial and final year

```

Time_Value_Truck = a_rst("Time_Value_Truck")
Time_Value_Pers = a_rst("Time_Value_Pers")
tt_ben_i = (ttsav_truck_i * Time_Value_Truck) + _
(ttsav_auto_i * AVO_auto * Time_Value_Pers)
tt_ben_f = (ttsav_truck_f * Time_Value_Truck) + _
(ttsav_auto_f * AVO_auto * Time_Value_Pers)

```

'Total Travel Time Calculations

```

m_TT_Sav = (tt_sav_i + tt_sav_f) * Forecast_Period / 2
m_TT_Min = m_TT_Sav * 60

```

```

G_tt_ben = (tt_ben_f - tt_ben_i) / (Forecast_Period - 1)
m_TT_Ben = (tt_ben_i * PofA) + (G_tt_ben * PofG)

```

'Total operating cost savings - initial and final year

```

op_cost_i = (ttsav_truck_i * OpCost_Truck) + _
(ttsav_auto_i * OpCost_Auto)
op_cost_f = (ttsav_truck_f * OpCost_Truck) + _
(ttsav_auto_f * OpCost_Auto)

```

'User Benefit Calculations

```

G_Opcost_Ben = (op_cost_f - op_cost_i) / (Forecast_Period - 1)
m_User_Ben = (op_cost_i * PofA) + (G_Opcost_Ben * PofG)

```

'Set module level values (to be displayed)

```

m_Values.Add m_TT_Sav, "TT_SAV"
m_Values.Add m_TT_Min, "TT_MIN"
m_Values.Add m_TT_Ben, "TT_BEN"
m_Values.Add m_User_Ben, "USER_BEN"

```

```

'Air Pollution Calculations - for diverted auto trips -
'none in current procedure, but keep as placeholder
  m_CO_Tons = 0
  m_VOC_Tons = 0
  m_NOX_Tons = 0
  m_PM10_Tons = 0
  m_Env_Ben = 0

'Set module level values (to be displayed)
  m_Values.Add m_CO_Tons, "CO_TONS"
  m_Values.Add m_VOC_Tons, "VOC_TONS"
  m_Values.Add m_NOX_Tons, "NOX_TONS"
  m_Values.Add m_PM10_Tons, "PM10_TONS"
  m_Values.Add m_Env_Ben, "ENV_BEN"

'Safety Calculations
  Fatality_Cost = a_rst("Fatality_Cost")
  Disable_Cost = a_rst("Disable_Cost")
  Evident_Cost = a_rst("Evident_Cost")
  Possible_Cost = a_rst("Possible_Cost")
  PDO_Cost = a_rst("PDO_Cost")

'Accident reduction factors
  R = 0
  RP = 0

  Dim whereSQL As String
  whereSQL = ReturnSafetyWhereSQL(m_rst)
  If whereSQL <> "" Then
    'Get 1-d array with values or zero for 5 improvement types
    Rarr = ReturnSafetyArray(whereSQL, "Fat_Inj_Red")
    RParr = ReturnSafetyArray(whereSQL, "PDO_Red")

    If IsArray(Rarr) Then
      For i = 0 To 4
        If i <> 0 Then
          'For fatality and injury accidents
          Rterm = Rterm * ((100 - Rarr(i - 1)) / 100)
          R = R + (Rarr(i) * Rterm)
          'For property damage accidents
          RPterm = RPterm * ((100 - RParr(i - 1)) / 100)
          RP = RP + (RParr(i) * Rterm)
        Else
          'For fatality and injury accidents
          R = Rarr(i)
          Rterm = 1
          'For property damage accidents
          RP = RParr(i)
          RPterm = 1
        End If
      Next
    End If
  End If

'Accident reduction due to proposed project

```

```

ann_fatality = m_rst("3yr_Fatality") * (R / 100) / 3
ann_disable = m_rst("3yr_Disable") * (R / 100) / 3
ann_evident = m_rst("3yr_Evident") * (R / 100) / 3
ann_possible = m_rst("3yr_Possible") * (R / 100) / 3
ann_property = m_rst("3yr_Property") * (RP / 100) / 3

```

```

m_Fatality = (-1) * ann_fatality * 20
m_Injury = (-1) * (ann_disable + ann_evident + ann_possible) * 20
m_Property = (-1) * ann_property * 20

```

'Calculate safety benefits

```

m_Safety_Ben = ((ann_fatality * Fatality_Cost) +
(ann_disable * Disable_Cost) + (ann_evident * Evident_Cost) +
(ann_possible * Possible_Cost) + (ann_property * PDO_Cost)) * PofA

```

'Set module level values (to be displayed)

```

m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

```

'Cost Calculations

'Capital Cost

For i = 1 To 5

```

m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))

```

Next

```

m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

```

'Operations and Maintenance Cost

```

m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

```

'Terminal cost

```

m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

```

'Total Costs

```

m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

```

```

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
If (m_Total_Cost - m_Terminal_Cost) = 0 Then
m_BCR = 0
Else
m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
End If
If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
m_WSDOT_BCR = 0
Else
m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
End If
'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"

```

```

m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

```

'Calculation for the System Operation and Maintenance
  m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
  m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
  m_Sys_Pres = 100 * m_rst("Q2A")
  m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
  m_Sp_Needs = 100 * m_rst("Q3A")
  m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
  Else
    m_Cong_Rel = (m_rst("Q4") * 50)
  End If
  m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
  Else
    m_Safety = m_rst("Q7B") * 50
  End If
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design

```

```

'The -1 Multiplication Corrects the negative sign introduced for true
m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
Else
    m_Collab = (m_rst("Q10A") * 50) + 50
End If
m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + m_rst("Q14A") * 50
Else
    m_Air_Qual = 0
End If
m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
    (33 * m_rst("Q15C"))
m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
    (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
    (1 + m_rst("Q16E") + m_rst("Q16F"))
m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
m_Resource = 100 * m_rst("Q17")
m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing

```

```
Set m_rst = Nothing  
Set m_dbs = Nothing  
Set m_Values = Nothing  
End Sub
```


High Occupancy Vehicle Lane

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim a_rst As DAO.Recordset

Dim m_qryName As String

Dim m_calcsComplete As Boolean

Dim m_Values As ValueCollection

'Variables used in intermediate calculations (variants)

Dim m_TT_Sav

Dim m_OpCost

Dim m_Approach

Dim m_WSDOT_Cap

Dim m_Federal_Cap

Dim m_Other1_Cap

Dim m_Other2_Cap

Dim m_Other3_Cap

Dim m_WSDOT_OM

Dim m_Federal_OM

Dim m_Other1_OM

Dim m_Other2_OM

Dim m_Other3_OM

Dim m_FishBarrier_Ben

Dim m_StormWater_Ben

Dim m_NoiseBarrier_Ben

Dim m_FishBarrier_Cap

Dim m_StormWater_Cap

Dim m_NoiseBarrier_Cap

'Variables to hold the calculated values (variants)

Dim m_Total_Benefit

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_EnvRetrofit_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Other1_Cost

Dim m_Other2_Cost

Dim m_Other3_Cost

Dim m_Cap_Cost

```
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
Private Sub Class_Initialize()
    m_qryName = "hwy_calc_HOV_Lane"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If
```

```

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
    Forecast_Period = 20

'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period) - (Forecast_Period / _
        ((1 + Discount_Rate) ^ Forecast_Period)))
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Calculate traffic distribution - exit if no curve was selected
'Travel Time Savings Calculations
    ADT_1 = m_rst("ADT_1")
    growth_rate = m_rst("GP_Growth")
    m_Length = m_rst("RoadLength")
    SOV_percent = m_rst("SOV_percent")
    HOV2_percent = m_rst("HOV2_percent")
    HOV3_percent = m_rst("HOV3_percent")

```

```

HOV4_percent = m_rst("HOV4_percent")
vanpool_percent = m_rst("vanpool_percent")
transit_percent = m_rst("transit_percent")
otherbus_percent = m_rst("otherbus_percent")
AVO_4carpool_1 = m_rst("AVO_4carpool_1")
AVO_4carpool_20 = m_rst("AVO_4carpool_20")
AVO_vanpool_1 = m_rst("AVO_vanpool_1")
AVO_vanpool_20 = m_rst("AVO_vanpool_20")
AVO_transit_1 = m_rst("AVO_transit_1")
AVO_transit_20 = m_rst("AVO_transit_20")
AVO_bus_1 = m_rst("AVO_bus_1")
AVO_bus_20 = m_rst("AVO_bus_20")

'Determine operating cost values
Auto_OpCostHr_Full = a_rst("Veh_OpCostHr_Full")
Auto_OpCostHr_Direct = a_rst("Veh_OpCostHr_Direct")
Truck_OpCostHr_Full = a_rst("Truck_OpCostHr_Full")
Truck_OpCostHr_Direct = a_rst("Truck_OpCostHr_Direct")
  If a_rst("Full_Cost") Then
    m_Approach = "Full cost"
    OpCost_Auto = Auto_OpCostHr_Full
    OpCost_Truck = Truck_OpCostHr_Full
  Else
    m_Approach = "Direct cost"
    OpCost_Auto = Auto_OpCostHr_Direct
    OpCost_Truck = Truck_OpCostHr_Direct
  End If

'Calculate year 20 volume
ADT_20 = (ADT_1 * Forecast_Period * growth_rate) + ADT_1

'Calculate roadway capacities
GPCap_NW_BC = m_rst("LnCap_NW_BC") * m_rst("NBWB_GP_Lns_BC")
GPCap_NW_PC = m_rst("LnCap_NW_PC") * m_rst("NBWB_GP_Lns_PC")
HOVCap_NW_PC = m_rst("LnCap_NW_PC") * m_rst("NBWB_HOV_Lns_PC")
GPCap_SE_BC = m_rst("LnCap_SE_BC") * m_rst("SBEB_GP_Lns_BC")
GPCap_SE_PC = m_rst("LnCap_SE_PC") * m_rst("SBEB_GP_Lns_PC")
HOVCap_SE_PC = m_rst("LnCap_SE_PC") * m_rst("SBEB_HOV_Lns_PC")

Dim crvid As Integer

If Not IsNumeric(m_rst("Curve_ID")) Then Exit Sub
crvid = m_rst("Curve_ID")
'these are the arrays that hold all values to be used later
Dim Dir1 As Variant
Dim Dir2 As Variant

Dim NW_ivol_i(24) As Variant
Dim NW_ivol_f(24) As Variant
Dim SE_ivol_i(24) As Variant
Dim SE_ivol_f(24) As Variant

Dim NWGP_ivol_i(24) As Variant
Dim NWHOV_ivol_i(24) As Variant
Dim NWGP_ivol_f(24) As Variant
Dim NWHOV_ivol_f(24) As Variant

```

```
Dim SEGP_ivol_i(24) As Variant
Dim SEHOV_ivol_i(24) As Variant
Dim SEGP_ivol_f(24) As Variant
Dim SEHOV_ivol_f(24) As Variant
```

```
Dim NWvol_bci(24) As Variant
Dim NWGPvol_pci(24) As Variant
Dim NWHOVvol_pci(24) As Variant
Dim NWvol_bcf(24) As Variant
Dim NWGPvol_pcf(24) As Variant
Dim NWHOVvol_pcf(24) As Variant
Dim SEvol_bci(24) As Variant
Dim SEGPvol_pci(24) As Variant
Dim SEHOVvol_pci(24) As Variant
Dim SEvol_bcf(24) As Variant
Dim SEGPvol_pcf(24) As Variant
Dim SEHOVvol_pcf(24) As Variant
```

```
Dim NW_fvol_bci(24) As Variant
Dim NWGP_fvol_pci(24) As Variant
Dim NWHOV_fvol_pci(24) As Variant
Dim NW_fvol_bcf(24) As Variant
Dim NWGP_fvol_pcf(24) As Variant
Dim NWHOV_fvol_pcf(24) As Variant
Dim SE_fvol_bci(24) As Variant
Dim SEGP_fvol_pci(24) As Variant
Dim SEHOV_fvol_pci(24) As Variant
Dim SE_fvol_bcf(24) As Variant
Dim SEGP_fvol_pcf(24) As Variant
Dim SEHOV_fvol_pcf(24) As Variant
```

```
Dim NW_ttsav_i(24) As Variant
Dim NW_ttsav_f(24) As Variant
Dim SE_ttsav_i(24) As Variant
Dim SE_ttsav_f(24) As Variant
```

```
Dir1 = ReturnSpeedCurveDirectionData(crvid, s_NB_WB)
Dir2 = ReturnSpeedCurveDirectionData(crvid, s_SB_EB)
```

```
'calculate sum of traffic in both directions
```

```
For hr = 1 To 24
```

```
    Twoway_sum = Twoway_sum + Dir1(hr - 1) + Dir2(hr - 1)
```

```
Next hr
```

```
'calculate initial hourly volumes - not adjusted for congestion
```

```
For hr = 1 To 24
```

```
    NW_ivol_i(hr - 1) = (Dir1(hr - 1) / Twoway_sum) * ADT_1
```

```
    NW_ivol_f(hr - 1) = (Dir1(hr - 1) / Twoway_sum) * ADT_20
```

```
    SE_ivol_i(hr - 1) = (Dir2(hr - 1) / Twoway_sum) * ADT_1
```

```
    SE_ivol_f(hr - 1) = (Dir2(hr - 1) / Twoway_sum) * ADT_20
```

```
Next hr
```

```
'allocate calculated volumes between GP and HOV
```

```
For hr = 1 To 24
```

```
    NWGP_ivol_i(hr - 1) = NW_ivol_i(hr - 1) * SOV_percent
```

```
    NWHOV_ivol_i(hr - 1) = NW_ivol_i(hr - 1) * (1 - SOV_percent)
```

```

NWGP_ivol_f(hr - 1) = NW_ivol_f(hr - 1) * SOV_percent
NWHOV_ivol_f(hr - 1) = NW_ivol_f(hr - 1) * (1 - SOV_percent)
SEGP_ivol_i(hr - 1) = SE_ivol_i(hr - 1) * SOV_percent
SEHOV_ivol_i(hr - 1) = SE_ivol_i(hr - 1) * (1 - SOV_percent)
SEGP_ivol_f(hr - 1) = SE_ivol_f(hr - 1) * SOV_percent
SEHOV_ivol_f(hr - 1) = SE_ivol_f(hr - 1) * (1 - SOV_percent)
Next hr

```

'Redistribute volumes for hours in which v/c exceeds 1.2

'Initialize excess volume variables

```

excess_nwGP_bci = 0
excess_nwGP_pci = 0
excess_nwHOV_pci = 0
excess_nwGP_bcf = 0
excess_nwGP_pcf = 0
excess_nwHOV_pcf = 0
excess_seGP_bci = 0
excess_seGP_pci = 0
excess_seHOV_pci = 0
excess_seGP_bcf = 0
excess_seGP_pcf = 0
excess_seHOV_pcf = 0

```

'For hours 18 to 24, excess hourly volume shifts to following hour

For hr = 18 To 24

```

NWvol_bci(hr - 1) = NW_ivol_i(hr - 1) + excess_nwGP_bci
If NWvol_bci(hr - 1) > (GPCap_NW_BC * 1.2) Then
  NW_fvol_bci(hr - 1) = (GPCap_NW_BC * 1.2)
  excess_nwGP_bci = NWvol_bci(hr - 1) - (GPCap_NW_BC * 1.2)
Else
  NW_fvol_bci(hr - 1) = NWvol_bci(hr - 1)
  excess_nwGP_bci = 0
End If

```

```

NWGPvol_pci(hr - 1) = NWGP_ivol_i(hr - 1) + excess_nwGP_pci
If NWGPvol_pci(hr - 1) > (GPCap_NW_PC * 1.2) Then
  NWGP_fvol_pci(hr - 1) = (GPCap_NW_PC * 1.2)
  excess_nwGP_pci = NWGPvol_pci(hr - 1) - (GPCap_NW_PC * 1.2)
Else
  NWGP_fvol_pci(hr - 1) = NWGPvol_pci(hr - 1)
  excess_nwGP_pci = 0
End If

```

```

NWHOVvol_pci(hr - 1) = NWHOV_ivol_i(hr - 1) + excess_nwHOV_pci
If NWHOVvol_pci(hr - 1) > (HOVCap_NW_PC * 1.2) Then
  NWHOV_fvol_pci(hr - 1) = (HOVCap_NW_PC * 1.2)
  excess_nwHOV_pci = NWHOVvol_pci(hr - 1) - (HOVCap_NW_PC * 1.2)
Else
  NWHOV_fvol_pci(hr - 1) = NWHOVvol_pci(hr - 1)
  excess_nwHOV_pci = 0
End If

```

```

NWvol_bcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nwGP_bcf
If NWvol_bcf(hr - 1) > (GPCap_NW_BC * 1.2) Then
  NW_fvol_bcf(hr - 1) = (GPCap_NW_BC * 1.2)
  excess_nwGP_bcf = NWvol_bcf(hr - 1) - (GPCap_NW_BC * 1.2)

```

```

Else
  NW_fvol_bcf(hr - 1) = NWvol_bcf(hr - 1)
  excess_nwGP_bcf = 0
End If

NWGPvol_pcf(hr - 1) = NWGP_ivol_f(hr - 1) + excess_nwGP_pcf
If NWGPvol_pcf(hr - 1) > (GPCap_NW_PC * 1.2) Then
  NWGP_fvol_pcf(hr - 1) = (GPCap_NW_PC * 1.2)
  excess_nwGP_pcf = NWGPvol_pcf(hr - 1) - (GPCap_NW_PC * 1.2)
Else
  NWGP_fvol_pcf(hr - 1) = NWGPvol_pcf(hr - 1)
  excess_nwGP_pcf = 0
End If

NWHOVvol_pcf(hr - 1) = NWHOV_ivol_f(hr - 1) + excess_nwHOV_pcf
If NWHOVvol_pcf(hr - 1) > (HOVCap_NW_PC * 1.2) Then
  NWHOV_fvol_pcf(hr - 1) = (HOVCap_NW_PC * 1.2)
  excess_nwHOV_pcf = NWHOVvol_pcf(hr - 1) - (HOVCap_NW_PC * 1.2)
Else
  NWHOV_fvol_pcf(hr - 1) = NWHOVvol_pcf(hr - 1)
  excess_nwHOV_pcf = 0
End If

SEvol_bci(hr - 1) = SE_ivol_i(hr - 1) + excess_seGP_bci
If SEvol_bci(hr - 1) > (GPCap_SE_BC * 1.2) Then
  SE_fvol_bci(hr - 1) = (GPCap_SE_BC * 1.2)
  excess_seGP_bci = NWvol_bci(hr - 1) - (GPCap_SE_BC * 1.2)
Else
  SE_fvol_bci(hr - 1) = SEvol_bci(hr - 1)
  excess_seGP_bci = 0
End If

SEGPvol_pci(hr - 1) = SEGP_ivol_i(hr - 1) + excess_seGP_pci
If SEGPvol_pci(hr - 1) > (GPCap_SE_PC * 1.2) Then
  SEGP_fvol_pci(hr - 1) = (GPCap_SE_PC * 1.2)
  excess_seGP_pci = SEGPvol_pci(hr - 1) - (GPCap_SE_PC * 1.2)
Else
  SEGP_fvol_pci(hr - 1) = SEGPvol_pci(hr - 1)
  excess_seGP_pci = 0
End If

SEHOVvol_pci(hr - 1) = SEHOV_ivol_i(hr - 1) + excess_seHOV_pci
If SEHOVvol_pci(hr - 1) > (HOVCap_SE_PC * 1.2) Then
  SEHOV_fvol_pci(hr - 1) = (HOVCap_SE_PC * 1.2)
  excess_seHOV_pci = SEHOVvol_pci(hr - 1) - (HOVCap_SE_PC * 1.2)
Else
  SEHOV_fvol_pci(hr - 1) = SEHOVvol_pci(hr - 1)
  excess_seHOV_pci = 0
End If

SEvol_bcf(hr - 1) = SE_ivol_f(hr - 1) + excess_seGP_bcf
If SEvol_bcf(hr - 1) > (GPCap_SE_BC * 1.2) Then
  SE_fvol_bcf(hr - 1) = (GPCap_SE_BC * 1.2)
  excess_seGP_bcf = SEvol_bcf(hr - 1) - (GPCap_SE_BC * 1.2)
Else
  SE_fvol_bcf(hr - 1) = SEvol_bcf(hr - 1)

```

```

    excess_seGP_bcf = 0
  End If

```

```

  SEGPvol_pcf(hr - 1) = SEGP_ivol_f(hr - 1) + excess_seGP_pcf
  If SEGPvol_pcf(hr - 1) > (GPCap_SE_PC * 1.2) Then
    SEGP_fvol_pcf(hr - 1) = (GPCap_SE_PC * 1.2)
    excess_seGP_pcf = SEGPvol_pcf(hr - 1) - (GPCap_SE_PC * 1.2)
  Else
    SEGP_fvol_pcf(hr - 1) = SEGPvol_pcf(hr - 1)
    excess_seGP_pcf = 0
  End If

```

```

  SEHOVvol_pcf(hr - 1) = SEHOV_ivol_f(hr - 1) + excess_seHOV_pcf
  If SEHOVvol_pcf(hr - 1) > (HOVCap_SE_PC * 1.2) Then
    SEHOV_fvol_pcf(hr - 1) = (HOVCap_SE_PC * 1.2)
    excess_seHOV_pcf = SEHOVvol_pcf(hr - 1) - (HOVCap_SE_PC * 1.2)
  Else
    SEHOV_fvol_pcf(hr - 1) = SEHOVvol_pcf(hr - 1)
    excess_seHOV_pcf = 0
  End If

```

```

Next hr

```

```

'For hours 1 to 17, excess hourly volume shifts to preceeding hour

```

```

'Reset excess volume variables

```

```

  excess_nwGP_bci = 0
  excess_nwGP_pci = 0
  excess_nwHOV_pci = 0
  excess_nwGP_bcf = 0
  excess_nwGP_pcf = 0
  excess_nwHOV_pcf = 0
  excess_seGP_bci = 0
  excess_seGP_pci = 0
  excess_seHOV_pci = 0
  excess_seGP_bcf = 0
  excess_seGP_pcf = 0
  excess_seHOV_pcf = 0

```

```

For hr = 17 To 1 Step -1

```

```

  NWvol_bci(hr - 1) = NW_ivol_i(hr - 1) + excess_nwGP_bci
  If NWvol_bci(hr - 1) > (GPCap_NW_BC * 1.2) Then
    NW_fvol_bci(hr - 1) = (GPCap_NW_BC * 1.2)
    excess_nwGP_bci = NWvol_bci(hr - 1) - (GPCap_NW_BC * 1.2)
  Else
    NW_fvol_bci(hr - 1) = NWvol_bci(hr - 1)
    excess_nwGP_bci = 0
  End If

```

```

  NWGPvol_pci(hr - 1) = NWGP_ivol_i(hr - 1) + excess_nwGP_pci
  If NWGPvol_pci(hr - 1) > (GPCap_NW_PC * 1.2) Then
    NWGP_fvol_pci(hr - 1) = (GPCap_NW_PC * 1.2)
    excess_nwGP_pci = NWGPvol_pci(hr - 1) - (GPCap_NW_PC * 1.2)
  Else
    NWGP_fvol_pci(hr - 1) = NWGPvol_pci(hr - 1)
    excess_nwGP_pci = 0
  End If

```



```

NWHOVvol_pci(hr - 1) = NWHOV_ivol_i(hr - 1) + excess_nwHOV_pci
If NWHOVvol_pci(hr - 1) > (HOVCap_NW_PC * 1.2) Then
  NWHOV_fvol_pci(hr - 1) = (HOVCap_NW_PC * 1.2)
  excess_nwHOV_pci = NWHOVvol_pci(hr - 1) - (HOVCap_NW_PC * 1.2)
Else
  NWHOV_fvol_pci(hr - 1) = NWHOVvol_pci(hr - 1)
  excess_nwHOV_pci = 0
End If

```

```

NWvol_bcf(hr - 1) = NW_ivol_f(hr - 1) + excess_nwGP_bcf
If NWvol_bcf(hr - 1) > (GPCap_NW_BC * 1.2) Then
  NW_fvol_bcf(hr - 1) = (GPCap_NW_BC * 1.2)
  excess_nwGP_bcf = NWvol_bcf(hr - 1) - (GPCap_NW_BC * 1.2)
Else
  NW_fvol_bcf(hr - 1) = NWvol_bcf(hr - 1)
  excess_nwGP_bcf = 0
End If

```

```

NWGPvol_pcf(hr - 1) = NWGP_ivol_f(hr - 1) + excess_nwGP_pcf
If NWGPvol_pcf(hr - 1) > (GPCap_NW_PC * 1.2) Then
  NWGP_fvol_pcf(hr - 1) = (GPCap_NW_PC * 1.2)
  excess_nwGP_pcf = NWGPvol_pcf(hr - 1) - (GPCap_NW_PC * 1.2)
Else
  NWGP_fvol_pcf(hr - 1) = NWGPvol_pcf(hr - 1)
  excess_nwGP_pcf = 0
End If

```

```

NWHOVvol_pcf(hr - 1) = NWHOV_ivol_f(hr - 1) + excess_nwHOV_pcf
If NWHOVvol_pcf(hr - 1) > (HOVCap_NW_PC * 1.2) Then
  NWHOV_fvol_pcf(hr - 1) = (HOVCap_NW_PC * 1.2)
  excess_nwHOV_pcf = NWHOVvol_pcf(hr - 1) - (HOVCap_NW_PC * 1.2)
Else
  NWHOV_fvol_pcf(hr - 1) = NWHOVvol_pcf(hr - 1)
  excess_nwHOV_pcf = 0
End If

```

```

SEvol_bci(hr - 1) = SE_ivol_i(hr - 1) + excess_seGP_bci
If SEvol_bci(hr - 1) > (GPCap_SE_BC * 1.2) Then
  SE_fvol_bci(hr - 1) = (GPCap_SE_BC * 1.2)
  excess_seGP_bci = SEvol_bci(hr - 1) - (GPCap_SE_BC * 1.2)
Else
  SE_fvol_bci(hr - 1) = SEvol_bci(hr - 1)
  excess_seGP_bci = 0
End If

```

```

SEGPvol_pci(hr - 1) = SEGP_ivol_i(hr - 1) + excess_seGP_pci
If SEGPvol_pci(hr - 1) > (GPCap_SE_PC * 1.2) Then
  SEGP_fvol_pci(hr - 1) = (GPCap_SE_PC * 1.2)
  excess_seGP_pci = SEGPvol_pci(hr - 1) - (GPCap_SE_PC * 1.2)
Else
  SEGP_fvol_pci(hr - 1) = SEGPvol_pci(hr - 1)
  excess_seGP_pci = 0
End If

```

```

SEHOVvol_pci(hr - 1) = SEHOV_ivol_i(hr - 1) + excess_seHOV_pci

```

```

If SEHOVvol_pci(hr - 1) > (HOVCap_SE_PC * 1.2) Then
  SEHOV_fvol_pci(hr - 1) = (HOVCap_SE_PC * 1.2)
  excess_seHOV_pci = SEHOVvol_pci(hr - 1) - (HOVCap_SE_PC * 1.2)
Else
  SEHOV_fvol_pci(hr - 1) = SEHOVvol_pci(hr - 1)
  excess_seHOV_pci = 0
End If

```

```

SEvol_bcf(hr - 1) = SE_ivol_f(hr - 1) + excess_seGP_bcf
If SEvol_bcf(hr - 1) > (GPCap_SE_BC * 1.2) Then
  SE_fvol_bcf(hr - 1) = (GPCap_SE_BC * 1.2)
  excess_seGP_bcf = SEvol_bcf(hr - 1) - (GPCap_SE_BC * 1.2)
Else
  SE_fvol_bcf(hr - 1) = SEvol_bcf(hr - 1)
  excess_seGP_bcf = 0
End If

```

```

SEGPvol_pcf(hr - 1) = SEGP_ivol_f(hr - 1) + excess_seGP_pcf
If SEGPvol_pcf(hr - 1) > (GPCap_SE_PC * 1.2) Then
  SEGP_fvol_pcf(hr - 1) = (GPCap_SE_PC * 1.2)
  excess_seGP_pcf = SEGPvol_pcf(hr - 1) - (GPCap_SE_PC * 1.2)
Else
  SEGP_fvol_pcf(hr - 1) = SEGPvol_pcf(hr - 1)
  excess_seGP_pcf = 0
End If

```

```

SEHOVvol_pcf(hr - 1) = SEHOV_ivol_f(hr - 1) + excess_seHOV_pcf
If SEHOVvol_pcf(hr - 1) > (HOVCap_SE_PC * 1.2) Then
  SEHOV_fvol_pcf(hr - 1) = (HOVCap_SE_PC * 1.2)
  excess_seHOV_pcf = SEHOVvol_pcf(hr - 1) - (HOVCap_SE_PC * 1.2)
Else
  SEHOV_fvol_pcf(hr - 1) = SEHOVvol_pcf(hr - 1)
  excess_seHOV_pcf = 0
End If

```

Next hr

'initiate the speed curve class

If IsNull(m_rst("facility_BC")) Or IsNull(m_rst("facility_PC")) Then Exit Sub

Dim spCurveBC As New SpeedCurve

Dim spCurvePC As New SpeedCurve

spCurveBC.SetHighwayClass (m_rst("facility_BC"))

spCurvePC.SetHighwayClass (m_rst("facility_PC"))

'calculate vc ratios, operating speed (from curves), and travel time savings

For hr = 1 To 24

```

NW_vc_bci = NW_fvol_bci(hr - 1) / GPCap_NW_BC
NWGP_vc_pci = NWGP_fvol_pci(hr - 1) / GPCap_NW_PC
NWHOV_vc_pci = NWHOV_fvol_pci(hr - 1) / HOVCap_NW_PC
NW_vc_bcf = NW_fvol_bcf(hr - 1) / GPCap_NW_BC
NWGP_vc_pcf = NWGP_fvol_pcf(hr - 1) / GPCap_NW_PC
NWHOV_vc_pcf = NWHOV_fvol_pcf(hr - 1) / HOVCap_NW_PC

```

```

SE_vc_bci = SE_fvol_bci(hr - 1) / GPCap_SE_BC

```

```

SEGP_vc_pci = SEGP_fvol_pci(hr - 1) / GPCap_SE_PC
SEHOV_vc_pci = SEHOV_fvol_pci(hr - 1) / HOVCap_SE_PC
SE_vc_bcf = SE_fvol_bcf(hr - 1) / GPCap_SE_BC
SEGP_vc_pcf = SEGP_fvol_pcf(hr - 1) / GPCap_SE_PC
SEHOV_vc_pcf = SEHOV_fvol_pcf(hr - 1) / HOVCap_SE_PC

```

```

NW_os_bci = spCurveBC.RetrieveSpeedData(NW_vc_bci)
NWGP_os_pci = spCurvePC.RetrieveSpeedData(NWGP_vc_pci)
NWHOV_os_pci = spCurvePC.RetrieveSpeedData(NWHOV_vc_pci)
NW_os_bcf = spCurveBC.RetrieveSpeedData(NW_vc_bcf)
NWGP_os_pcf = spCurvePC.RetrieveSpeedData(NWGP_vc_pcf)
NWHOV_os_pcf = spCurvePC.RetrieveSpeedData(NWHOV_vc_pcf)
SE_os_bci = spCurveBC.RetrieveSpeedData(SE_vc_bci)
SEGP_os_pci = spCurvePC.RetrieveSpeedData(SEGP_vc_pci)
SEHOV_os_pci = spCurvePC.RetrieveSpeedData(SEHOV_vc_pci)
SE_os_bcf = spCurveBC.RetrieveSpeedData(SE_vc_bcf)
SEGP_os_pcf = spCurvePC.RetrieveSpeedData(SEGP_vc_pcf)
SEHOV_os_pcf = spCurvePC.RetrieveSpeedData(SEHOV_vc_pcf)

```

```

NW_ttsav_i(hr - 1) = m_Length * ((NW_fvol_bci(hr - 1) / NW_os_bci) - _
    (NWGP_fvol_pci(hr - 1) / NWGP_os_pci + NWHOV_fvol_pci(hr - 1) /
NWHOV_os_pci))
NW_ttsav_f(hr - 1) = m_Length * ((NW_fvol_bcf(hr - 1) / NW_os_bcf) - _
    (NWGP_fvol_pcf(hr - 1) / NWGP_os_pcf + NWHOV_fvol_pcf(hr - 1) /
NWHOV_os_pcf))
SE_ttsav_i(hr - 1) = m_Length * ((SE_fvol_bci(hr - 1) / SE_os_bci) - _
    (SEGP_fvol_pci(hr - 1) / SEGP_os_pci + SEHOV_fvol_pci(hr - 1) / SEHOV_os_pci))
SE_ttsav_f(hr - 1) = m_Length * ((SE_fvol_bcf(hr - 1) / SE_os_bcf) - _
    (SEGP_fvol_pcf(hr - 1) / SEGP_os_pcf + SEHOV_fvol_pcf(hr - 1) / SEHOV_os_pcf))

```

'Total travel time savings - initial and final year

```

TT_Sav_i = TT_Sav_i + NW_ttsav_i(hr - 1) + SE_ttsav_i(hr - 1)
TT_Sav_f = TT_Sav_f + NW_ttsav_f(hr - 1) + SE_ttsav_f(hr - 1)

```

Next hr

```

Set spCurveBC = Nothing
Set spCurvePC = Nothing

```

'Allot travel time savings between truck and non-truck

```

Truckper_i = m_rst("Truckper_1")
Truckper_f = m_rst("Truckper_20")
Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
ttsav_truck_i = TT_Sav_i * (SOV_percent * Truckper_i) * Ann_Daily_Benefit
ttsav_truck_f = TT_Sav_f * (SOV_percent * Truckper_f) * Ann_Daily_Benefit
ttsav_veh_i = TT_Sav_i * (1 - (SOV_percent * Truckper_i)) * Ann_Daily_Benefit
ttsav_veh_f = TT_Sav_f * (1 - (SOV_percent * Truckper_f)) * Ann_Daily_Benefit

```

'Person travel time savings

```

ttsav_per_i = ttsav_veh_i * ((1 - (SOV_percent * Truckper_i) * 1) + _
    (HOV2_percent * 2) + (HOV3_percent * 3) + (HOV4_percent * AVO_4carpool_1) + _
    (vanpool_percent * AVO_vanpool_1) + (transit_percent * AVO_transit_1) + _
    (otherbus_percent * AVO_bus_1))

```

```

ttsav_per_f = ttsav_veh_f * ((1 - (SOV_percent * Truckper_f) * 1) + _
    (HOV2_percent * 2) + (HOV3_percent * 3) + (HOV4_percent * AVO_4carpool_20) + _

```

```
(vanpool_percent * AVO_vanpool_20) + (transit_percent * AVO_transit_20) + _
(otherbus_percent * AVO_bus_20))
```

```
'Total travel time benefit - initial and final year
Time_Value_Truck = a_rst("Time_Value_Truck")
Time_Value_Pers = a_rst("Time_Value_Pers")
tt_ben_i = (ttsav_truck_i * Time_Value_Truck) + _
           (ttsav_per_i * Time_Value_Pers)
tt_ben_f = (ttsav_truck_f * Time_Value_Truck) + _
           (ttsav_per_f * Time_Value_Pers)
```

```
'Total Travel Time Calculations
m_TT_Sav = (TT_Sav_i + TT_Sav_f) * Forecast_Period / 2
m_TT_Min = m_TT_Sav * 60
```

```
G_tt_ben = (tt_ben_f - tt_ben_i) / (Forecast_Period - 1)
m_TT_Ben = (tt_ben_i * PofA) + (G_tt_ben * PofG)
```

```
'Total operating cost savings - initial and final year
```

```
op_cost_i = (ttsav_truck_i * OpCost_Truck) + _
            (ttsav_veh_i * OpCost_Auto)
op_cost_f = (ttsav_truck_f * OpCost_Truck) + _
            (ttsav_veh_f * OpCost_Auto)
```

```
'User Benefit Calculations
```

```
G_Opcost_Ben = (op_cost_f - op_cost_i) / (Forecast_Period - 1)
m_User_Ben = (op_cost_i * PofA) + (G_Opcost_Ben * PofG)
```

```
'Set module level values (to be displayed)
```

```
m_Values.Add m_TT_Sav, "TT_SAV"
m_Values.Add m_TT_Min, "TT_MIN"
m_Values.Add m_TT_Ben, "TT_BEN"
m_Values.Add m_User_Ben, "USER_BEN"
```

```
'Air Pollution Calculations - for diverted auto trips -
```

```
'none in current procedure, but keep as placeholder
```

```
m_CO_Tons = 0
m_VOC_Tons = 0
m_NOX_Tons = 0
m_PM10_Tons = 0
m_Env_Ben = 0
```

```
'Set module level values (to be displayed)
```

```
m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"
m_Values.Add m_PM10_Tons, "PM10_TONS"
m_Values.Add m_Env_Ben, "ENV_BEN"
```

```
'Safety Calculations
```

```
Fatality_Cost = a_rst("Fatality_Cost")
Disable_Cost = a_rst("Disable_Cost")
Evident_Cost = a_rst("Evident_Cost")
Possible_Cost = a_rst("Possible_Cost")
PDO_Cost = a_rst("PDO_Cost")
```

```
'Accident reduction factors
```

```
R = 0
RP = 0
```

```
Dim whereSQL As String
```

```
whereSQL = ReturnSafetyWhereSQL(m_rst)
```

```
If whereSQL <> "" Then
```

```
    'Get 1-d array with values or zero for 5 improvement types
```

```
    Rarr = ReturnSafetyArray(whereSQL, "Fat_Inj_Red")
```

```
    RParr = ReturnSafetyArray(whereSQL, "PDO_Red")
```

```
    If IsArray(Rarr) Then
```

```
        For i = 0 To 4
```

```
            If i <> 0 Then
```

```
                'For fatality and injury accidents
```

```
                Rterm = Rterm * ((100 - Rarr(i - 1)) / 100)
```

```
                R = R + (Rarr(i) * Rterm)
```

```
                'For property damage accidents
```

```
                RPterm = RPterm * ((100 - RParr(i - 1)) / 100)
```

```
                RP = RP + (RParr(i) * RPterm)
```

```
            Else
```

```
                'For fatality and injury accidents
```

```
                R = Rarr(i)
```

```
                Rterm = 1
```

```
                'For property damage accidents
```

```
                RP = RParr(i)
```

```
                RPterm = 1
```

```
            End If
```

```
        Next
```

```
    End If
```

```
End If
```

```
'Accident reduction due to proposed project
```

```
ann_fatality = m_rst("3yr_Fatality") * (R / 100) / 3
```

```
ann_disable = m_rst("3yr_Disable") * (R / 100) / 3
```

```
ann_evident = m_rst("3yr_Evident") * (R / 100) / 3
```

```
ann_possible = m_rst("3yr_Possible") * (R / 100) / 3
```

```
ann_property = m_rst("3yr_Property") * (RP / 100) / 3
```

```
m_Fatality = (-1) * ann_fatality * 20
```

```
m_Injury = (-1) * (ann_disable + ann_evident + ann_possible) * 20
```

```
m_Property = (-1) * ann_property * 20
```

```
'Calculate safety benefits
```

```
m_Safety_Ben = ((ann_fatality * Fatality_Cost) +
(ann_disable * Disable_Cost) + (ann_evident * Evident_Cost) +
(ann_possible * Possible_Cost) + (ann_property * PDO_Cost)) * PofA
```

```
'Set module level values (to be displayed)
```

```
m_Values.Add m_Fatality, "FATALITY"
```

```
m_Values.Add m_Injury, "INJURY"
```

```
m_Values.Add m_Property, "PROPERTY"
```

```
m_Values.Add m_Safety_Ben, "SAFETY_BEN"
```

```
'Cost Calculations
```

```

'Capital Cost
For i = 1 To 5
  m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
  m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

```

```

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
If (m_Total_Cost - m_Terminal_Cost) = 0 Then
    m_BCR = 0
Else
    m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
End If
If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
    m_WSDOT_BCR = 0
Else
    m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
End If
'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

End Sub

Private Sub CalculateOutObjScores()

```

On Error Resume Next

```

'Calculation for the System Operation and Maintenance
  m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
  m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
  m_Sys_Pres = 100 * m_rst("Q2A")
  m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
  m_Sp_Needs = 100 * m_rst("Q3A")
  m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
  Else
    m_Cong_Rel = (m_rst("Q4") * 50)
  End If
  m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
  Else
    m_Safety = m_rst("Q7B") * 50
  End If
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  'The -1 Multiplication Corrects the negative sign introduced for true
  m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B"))) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E"))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
  Else
    m_Collab = (m_rst("Q10A") * 50) + 50
  End If
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality

```



```

    If m_rst("Q14B") = 1 Then
        m_Air_Qual = 50 + m_rst("Q14A") * 50
    Else
        m_Air_Qual = 0
    End If
    m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub

```

Interchange

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim a_rst As DAO.Recordset

Dim m_qryName As String

Dim m_calcsComplete As Boolean

Dim m_Values As ValueCollection

'Variables used in intermediate calculations (variants)

Dim m_TT_Sav

Dim m_OpCost

Dim m_Approach

Dim m_WSDOT_Cap

Dim m_Federal_Cap

Dim m_Other1_Cap

Dim m_Other2_Cap

Dim m_Other3_Cap

Dim m_WSDOT_OM

Dim m_Federal_OM

Dim m_Other1_OM

Dim m_Other2_OM

Dim m_Other3_OM

Dim m_FishBarrier_Ben

Dim m_StormWater_Ben

Dim m_NoiseBarrier_Ben

Dim m_FishBarrier_Cap

Dim m_StormWater_Cap

Dim m_NoiseBarrier_Cap

'Variables to hold the calculated values (variants)

Dim m_Total_Benefit

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_EnvRetrofit_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Other1_Cost

Dim m_Other2_Cost

Dim m_Other3_Cost

Dim m_Cap_Cost

```
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
Private Sub Class_Initialize()
    m_qryName = "hwy_calc_Interchange"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If
```

```

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
    On Error Resume Next
    'Calculate Forecast Period
    Forecast_Period = 20

    'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period) - (Forecast_Period / _
        ((1 + Discount_Rate) ^ Forecast_Period)))
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

    'Determine operating cost values
    Auto_OpCostHr_Full = a_rst("Veh_OpCostHr_Full")
    Auto_OpCostHr_Direct = a_rst("Veh_OpCostHr_Direct")
    Truck_OpCostHr_Full = a_rst("Truck_OpCostHr_Full")
    Truck_OpCostHr_Direct = a_rst("Truck_OpCostHr_Direct")
    If a_rst("Full_Cost") Then
        m_Approach = "Full cost"
        OpCost_Auto = Auto_OpCostHr_Full
    
```

```

    OpCost_Truck = Truck_OpCostHr_Full
Else
    m_Approach = "Direct cost"
    OpCost_Auto = Auto_OpCostHr_Direct
    OpCost_Truck = Truck_OpCostHr_Direct
End If

Dim crvid As Integer

If Not IsNumeric(m_rst("Curve_ID")) Then Exit Sub
crvid = m_rst("Curve_ID")
'these are the arrays that hold all values to be used later
Dim Dir1 As Variant
Dim Dir2 As Variant

Dim NW_per(24) As Variant
Dim NWpki_per(24) As Variant
Dim NWpkf_per(24) As Variant
Dim SE_per(24) As Variant
Dim SEpki_per(24) As Variant
Dim SEpkf_per(24) As Variant
Dim Both_per(24) As Variant

'Calculate AM peak hour percentages

AMpk_beg_i = m_rst("AMpk_beg_i")
AMpk_end_i = m_rst("AMpk_end_i")
AMpk_beg_f = m_rst("AMpk_beg_f")
AMpk_end_f = m_rst("AMpk_end_f")

AM_Max_i = 0
AM_Max_f = 0

For hr = 1 To 12
    NW_per(hr - 1) = Dir1(hr - 1)
    SE_per(hr - 1) = Dir2(hr - 1)
    Both_per(hr - 1) = NW_per(hr - 1) + SE_per(hr - 1)

    If hr < AMpeak_begin_i Then
        NWpki_per(hr - 1) = 0
        SEpki_per(hr - 1) = 0
    Else
        If hr > AMpeak_end_i Then
            NWpki_per(hr - 1) = 0
            SEpki_per(hr - 1) = 0
        Else
            NWpki_per(hr - 1) = NW_per(hr - 1)
            SEpki_per(hr - 1) = SE_per(hr - 1)
        End If
    End If

    peak_per_day_i = peak_per_day_i + NWpki_per(hr - 1) + SEpki_per(hr - 1)

    If AM_Max_i < Both_per(hr - 1) Then
        AM_Max_i = Both_per(hr - 1)
    End If

```

```

If hr < AMpeak_begin_f Then
  NWpkf_per(hr - 1) = 0
  SEpkf_per(hr - 1) = 0
Else
  If hr > AMpeak_end_i Then
    NWpkf_per(hr - 1) = 0
    SEpkf_per(hr - 1) = 0
  Else
    NWpkf_per(hr - 1) = NW_per(hr - 1)
    SEpkf_per(hr - 1) = SE_per(hr - 1)
  End If
End If

peak_per_day_f = peak_per_day_f + NWpkf_per(hr - 1) + SEpkf_per(hr - 1)

If AM_Max_f < Both_per(hr - 1) Then
  AM_Max_f = Both_per(hr - 1)
End If

Next hr

'Calculate PM peak hour percentages

PMpk_beg_i = m_rst("PMpk_beg_i")
PMpk_end_i = m_rst("PMpk_end_i")
PMpk_beg_f = m_rst("PMpk_beg_f")
PMpk_end_f = m_rst("PMpk_end_f")

PM_Max_i = 0
PM_Max_f = 0

For hr = 13 To 24
  NW_per(hr - 1) = Dir1(hr - 1)
  SE_per(hr - 1) = Dir2(hr - 1)
  Both_per(hr - 1) = NW_per(hr - 1) + SE_per(hr - 1)

  If hr < PMpeak_begin_i Then
    NWpki_per(hr - 1) = 0
    SEpki_per(hr - 1) = 0
  Else
    If hr > PMpeak_end_i Then
      NWpki_per(hr - 1) = 0
      SEpki_per(hr - 1) = 0
    Else
      NWpki_per(hr - 1) = NW_per(hr - 1)
      SEpki_per(hr - 1) = SE_per(hr - 1)
    End If
  End If

  peak_per_day_i = peak_per_day_i + NWpki_per(hr - 1) + SEpki_per(hr - 1)

  If PM_Max_i < Both_per(hr - 1) Then
    PM_Max_i = Both_per(hr - 1)
  End If
End For

```

```

If hr < PMpeak_begin_f Then
  NWpkf_per(hr - 1) = 0
  SEpkf_per(hr - 1) = 0
Else
  If hr > PMpeak_end_i Then
    NWpkf_per(hr - 1) = 0
    SEpkf_per(hr - 1) = 0
  Else
    NWpkf_per(hr - 1) = NW_per(hr - 1)
    SEpkf_per(hr - 1) = SE_per(hr - 1)
  End If
End If

peak_per_day_f = peak_per_day_f + NWpkf_per(hr - 1) + SEpkf_per(hr - 1)

If PM_Max_f < Both_per(hr - 1) Then
  PM_Max_f = Both_per(hr - 1)
End If

Next hr

'Calculate the Daily / Peak Hour Benefit Ratio

If AM_Max_i > PM_Max_i Then
  Day_Max_i = AM_Max_i
Else
  Day_Max_i = PM_Max_i
End If

If AM_Max_f > PM_Max_f Then
  Day_Max_f = AM_Max_f
Else
  Day_Max_f = PM_Max_f
End If

Day_PkHr_Rat_i = Day_Max_i / peak_per_day_i
Day_PkHr_Rat_f = Day_Max_f / peak_per_day_f

'Calculate travel time savings

'Daily Volumes
cb_DayVol_i = cb_vol_1 / Kfactor
cb_DayVol_f = cb_vol_20 / Kfactor
bc_DayVol_i = bc_vol_1 / Kfactor
bc_DayVol_f = bc_vol_20 / Kfactor
ca_DayVol_i = ca_vol_1 / Kfactor
ca_DayVol_f = ca_vol_20 / Kfactor
ac_DayVol_i = ac_vol_1 / Kfactor
ac_DayVol_f = ac_vol_20 / Kfactor

'Travel Times
cb_TT_BC = (m_rst("c1_b1_dist_BC") / m_rst("c1_b1_speed_BC")) + _
(m_rst("c2_b2_dist_BC") / m_rst("c2_b2_speed_BC")) + _
(m_rst("c3_b3_dist_BC") / m_rst("c3_b3_speed_BC"))
cb_TT_PC = (m_rst("c1_b1_dist_PC") / m_rst("c1_b1_speed_PC")) + _
(m_rst("c2_b2_dist_PC") / m_rst("c2_b2_speed_PC")) + _

```

```

(m_rst("c3_b3_dist_PC") / m_rst("c3_b3_speed_PC"))
bc_TT_BC = (m_rst("b1_c1_dist_BC") / m_rst("b1_c1_speed_BC")) + _
(m_rst("b2_c2_dist_BC") / m_rst("b2_c2_speed_BC")) + _
(m_rst("b3_c3_dist_BC") / m_rst("b3_c3_speed_BC")) + _
(m_rst("b4_c4_dist_BC") / m_rst("b4_c4_speed_BC"))
bc_TT_PC = (m_rst("b1_c1_dist_PC") / m_rst("b1_c1_speed_PC")) + _
(m_rst("b2_c2_dist_PC") / m_rst("b2_c2_speed_PC")) + _
(m_rst("b3_c3_dist_PC") / m_rst("b3_c3_speed_PC")) + _
(m_rst("b4_c4_dist_PC") / m_rst("b4_c4_speed_PC"))
ca_TT_BC = (m_rst("c1_a1_dist_BC") / m_rst("c1_a1_speed_BC")) + _
(m_rst("c2_a2_dist_BC") / m_rst("c2_a2_speed_BC")) + _
(m_rst("c3_a3_dist_BC") / m_rst("c3_a3_speed_BC")) + _
(m_rst("c4_a4_dist_BC") / m_rst("c4_a4_speed_BC")) + _
(m_rst("c5_a5_dist_BC") / m_rst("c5_a5_speed_BC")) + _
ca_TT_PC = (m_rst("c1_a1_dist_PC") / m_rst("c1_a1_speed_PC")) + _
(m_rst("c2_a2_dist_PC") / m_rst("c2_a2_speed_PC")) + _
(m_rst("c3_a3_dist_PC") / m_rst("c3_a3_speed_PC"))
ac_TT_BC = (m_rst("a1_c1_dist_BC") / m_rst("a1_c1_speed_BC")) + _
(m_rst("a2_c2_dist_BC") / m_rst("a2_c2_speed_BC")) + _
(m_rst("a3_c3_dist_BC") / m_rst("a3_c3_speed_BC")) + _
(m_rst("a4_c4_dist_BC") / m_rst("a4_c4_speed_BC"))
ac_TT_PC = (m_rst("a1_c1_dist_PC") / m_rst("a1_c1_speed_PC")) + _
(m_rst("a2_c2_dist_PC") / m_rst("a2_c2_speed_PC"))

```

'Travel Time Savings

```

cb_TTsav_i = cb_DayVol_i * (cb_TT_BC - cb_TT_PC)
cb_TTsav_f = cb_DayVol_f * (cb_TT_BC - cb_TT_PC)
bc_TTsav_i = bc_DayVol_i * (bc_TT_BC - bc_TT_PC)
bc_TTsav_f = bc_DayVol_f * (bc_TT_BC - bc_TT_PC)
ca_TTsav_i = ca_DayVol_i * (ca_TT_BC - ca_TT_PC)
ca_TTsav_f = ca_DayVol_f * (ca_TT_BC - ca_TT_PC)
ac_TTsav_i = ac_DayVol_i * (ac_TT_BC - ac_TT_PC)
ac_TTsav_f = ac_DayVol_f * (ac_TT_BC - ac_TT_PC)

TT_daysav_i = cb_TTsav_i + bc_TTsav_i + ca_TTsav_i + ac_TTsav_i
TT_daysav_f = cb_TTsav_f + bc_TTsav_f + ca_TTsav_f + ac_TTsav_f

```

'Convert tt savings from hourly to daily and allot between truck and auto

```

AVO_auto_pk = m_rst("AVO_auto_pk")
AVO_auto_npk = m_rst("AVO_auto_npk")
Truck_per = m_rst("Truck_per")
Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
Auto_per = 1 - Truck_per
nonpeak_per_day_i = 1 - peak_per_day_i
nonpeak_per_day_f = 1 - peak_per_day_f

TT_GPdaysav_i = TT_daysav_i * (1 - Truck_per)
TT_GPdaysav_f = TT_daysav_f * (1 - Truck_per)
TT_TRCKdaysav_i = TT_daysav_i * Truck_per
TT_TRCKdaysav_f = TT_daysav_f * Truck_per

ttsav_auto_i = TT_GPdaysav_i * (AVO_auto_npk * nonpeak_per_day_i + _
AVO_auto_pk * peak_per_day_i) * Day_PkHr_Rat_i * Ann_Daily_Benefit
ttsav_auto_f = TT_GPdaysav_f * (AVO_auto_npk * nonpeak_per_day_f + _
AVO_auto_pk * peak_per_day_f) * Day_PkHr_Rat_f * Ann_Daily_Benefit
ttsav_truck_i = TT_TRCKdaysav_i * Day_PkHr_Rat_i * Ann_Daily_Benefit

```



```
ttsav_truck_f = TT_TRCKdaysav_f * Day_PkHr_Rat_f * Ann_Daily_Benefit
```

```
'Total travel time benefit - initial and final year
```

```
Time_Value_Truck = a_rst("Time_Value_Truck")
```

```
Time_Value_Pers = a_rst("Time_Value_Pers")
```

```
tt_ben_i = (ttsav_truck_i * Time_Value_Truck) + _  
          (ttsav_auto_i * AVO_auto * Time_Value_Pers)
```

```
tt_ben_f = (ttsav_truck_f * Time_Value_Truck) + _  
          (ttsav_auto_f * AVO_auto * Time_Value_Pers)
```

```
'Total Travel Time Calculations
```

```
m_TT_Sav = (ttsav_auto_i + ttsav_auto_f + ttsav_truck_i + ttsav_truck_f) * _  
          Forecast_Period / 2
```

```
m_TT_Min = m_TT_Sav * 60
```

```
G_tt_ben = (tt_ben_f - tt_ben_i) / (Forecast_Period - 1)
```

```
m_TT_Ben = (tt_ben_i * PofA) + (G_tt_ben * PofG)
```

```
'Total operating cost savings - initial and final year
```

```
op_cost_i = (ttsav_truck_i * OpCost_Truck) + _  
          (ttsav_auto_i * OpCost_Auto)
```

```
op_cost_f = (ttsav_truck_f * OpCost_Truck) + _  
          (ttsav_auto_f * OpCost_Auto)
```

```
'User Benefit Calculations
```

```
G_Opcost_Ben = (op_cost_f - op_cost_i) / (Forecast_Period - 1)
```

```
m_User_Ben = (op_cost_i * PofA) + (G_Opcost_Ben * PofG)
```

```
'Set module level values (to be displayed)
```

```
m_Values.Add m_TT_Sav, "TT_SAV"
```

```
m_Values.Add m_TT_Min, "TT_MIN"
```

```
m_Values.Add m_TT_Ben, "TT_BEN"
```

```
m_Values.Add m_User_Ben, "USER_BEN"
```

```
'Air Pollution Calculations - for diverted auto trips -
```

```
'none in current procedure, but keep as placeholder
```

```
m_CO_Tons = 0
```

```
m_VOC_Tons = 0
```

```
m_NOX_Tons = 0
```

```
m_PM10_Tons = 0
```

```
m_Env_Ben = 0
```

```
'Set module level values (to be displayed)
```

```
m_Values.Add m_CO_Tons, "CO_TONS"
```

```
m_Values.Add m_VOC_Tons, "VOC_TONS"
```

```
m_Values.Add m_NOX_Tons, "NOX_TONS"
```

```
m_Values.Add m_PM10_Tons, "PM10_TONS"
```

```
m_Values.Add m_Env_Ben, "ENV_BEN"
```

```
'Safety Calculations
```

```
Fatality_Cost = a_rst("Fatality_Cost")
```

```
Disable_Cost = a_rst("Disable_Cost")
```

```
Evident_Cost = a_rst("Evident_Cost")
```

```
Possible_Cost = a_rst("Possible_Cost")
```

```
PDO_Cost = a_rst("PDO_Cost")
```

```

'Accident reduction factors
R = 0
RP = 0

Dim whereSQL As String
whereSQL = ReturnSafetyWhereSQL(m_rst)
If whereSQL <> "" Then
    'Get 1-d array with values or zero for 5 improvement types
    Rarr = ReturnSafetyArray(whereSQL, "Fat_Inj_Red")
    RParr = ReturnSafetyArray(whereSQL, "PDO_Red")

    If IsArray(Rarr) Then
        For i = 0 To 4
            If i <> 0 Then
                'For fatality and injury accidents
                Rterm = Rterm * ((100 - Rarr(i - 1)) / 100)
                R = R + (Rarr(i) * Rterm)
                'For property damage accidents
                RPterm = RPterm * ((100 - RParr(i - 1)) / 100)
                RP = RP + (RParr(i) * Rterm)
            Else
                'For fatality and injury accidents
                R = Rarr(i)
                Rterm = 1
                'For property damage accidents
                RP = RParr(i)
                RPterm = 1
            End If
        Next
    End If
End If

'Accident reduction due to proposed project
ann_fatality = m_rst("3yr_Fatality") * (R / 100) / 3
ann_disable = m_rst("3yr_Disable") * (R / 100) / 3
ann_evident = m_rst("3yr_Evident") * (R / 100) / 3
ann_possible = m_rst("3yr_Possible") * (R / 100) / 3
ann_property = m_rst("3yr_Property") * (RP / 100) / 3

m_Fatality = (-1) * ann_fatality * 20
m_Injury = (-1) * (ann_disable + ann_evident + ann_possible) * 20
m_Property = (-1) * ann_property * 20

'Calculate safety benefits
m_Safety_Ben = ((ann_fatality * Fatality_Cost) +
(ann_disable * Disable_Cost) + (ann_evident * Evident_Cost) +
(ann_possible * Possible_Cost) + (ann_property * PDO_Cost)) * PofA

'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations

```

```

'Capital Cost
For i = 1 To 5
  m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
  m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

```

```

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
If (m_Total_Cost - m_Terminal_Cost) = 0 Then
  m_BCR = 0
Else
  m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
End If
If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
  m_WSDOT_BCR = 0
Else
  m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
End If
'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

End Sub

Private Sub CalculateOutObjScores()

```

On Error Resume Next

```

'Calculation for the System Operation and Maintenance
  m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
  m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
  m_Sys_Pres = 100 * m_rst("Q2A")
  m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
  m_Sp_Needs = 100 * m_rst("Q3A")
  m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
  Else
    m_Cong_Rel = (m_rst("Q4") * 50)
  End If
  m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
  Else
    m_Safety = m_rst("Q7B") * 50
  End If
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  'The -1 Multiplication Corrects the negative sign introduced for true
  m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
  Else
    m_Collab = (m_rst("Q10A") * 50) + 50
  End If
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality

```

```

    If m_rst("Q14B") = 1 Then
        m_Air_Qual = 50 + m_rst("Q14A") * 50
    Else
        m_Air_Qual = 0
    End If
    m_Values.Add m_Air_Qual, "AIR_QUAL"
    'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
    'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
    'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub

```

Intersection

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim a_rst As DAO.Recordset

Dim m_qryName As String

Dim m_calcsComplete As Boolean

Dim m_Values As ValueCollection

'Variables used in intermediate calculations (variants)

Dim m_TT_Sav

Dim m_OpCost

Dim m_Approach

Dim m_WSDOT_Cap

Dim m_Federal_Cap

Dim m_Other1_Cap

Dim m_Other2_Cap

Dim m_Other3_Cap

Dim m_WSDOT_OM

Dim m_Federal_OM

Dim m_Other1_OM

Dim m_Other2_OM

Dim m_Other3_OM

Dim m_FishBarrier_Ben

Dim m_StormWater_Ben

Dim m_NoiseBarrier_Ben

Dim m_FishBarrier_Cap

Dim m_StormWater_Cap

Dim m_NoiseBarrier_Cap

'Variables to hold the calculated values (variants)

Dim m_Total_Benefit

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_EnvRetrofit_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Other1_Cost

Dim m_Other2_Cost

Dim m_Other3_Cost

Dim m_Cap_Cost

```
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
Private Sub Class_Initialize()
    m_qryName = "hwy_calc_Intersection"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If
```



```

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
    Forecast_Period = 20

'Determine operating cost values
    Auto_OpCostHr_Full = a_rst("Veh_OpCostHr_Full")
    Auto_OpCostHr_Direct = a_rst("Veh_OpCostHr_Direct")
    Truck_OpCostHr_Full = a_rst("Truck_OpCostHr_Full")
    Truck_OpCostHr_Direct = a_rst("Truck_OpCostHr_Direct")
    If a_rst("Full_Cost") Then
        m_Approach = "Full cost"
        OpCost_Auto = Auto_OpCostHr_Full
        OpCost_Truck = Truck_OpCostHr_Full
    Else
        m_Approach = "Direct cost"
        OpCost_Auto = Auto_OpCostHr_Direct
        OpCost_Truck = Truck_OpCostHr_Direct
    End If

'Calculate economic analysis factors

```

```

Discount_Rate = a_rst("Discount_Rate")
PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
  (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / _
  ((1 + Discount_Rate) ^ Forecast_Period)))
PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

```

'Traffic Variables

```

Vol_growth_rate = m_rst("Vol_growth_rate")
Truck_per = m_rst("Truck_Per")
Init_Year = m_rst("Init_Year")
Data_Year = m_rst("Data_Year")
Tot_vol_1_BC = m_rst("Tot_vol_1_BC")
Tot_vol_1_PC = m_rst("Tot_vol_1_PC")
Lanes_BC = m_rst("Lanes_BC")
Lanes_PC = m_rst("Lanes_PC")
Delay_BC = m_rst("Delay_BC")
Delay_PC = m_rst("Delay_PC")
VC_1_BC = m_rst("VC_1_BC")
VC_1_PC = m_rst("VC_1_PC")
App1_red_PC = m_rst("App1_red_PC")
App2_red_PC = m_rst("App2_red_PC")
App3_red_PC = m_rst("App3_red_PC")
App4_red_PC = m_rst("App4_red_PC")

```

'24 hour volume arrays

```

Dim uvol_BCI1(24) As Variant
Dim uvol_BCI2(24) As Variant
Dim uvol_BCI3(24) As Variant
Dim uvol_BCI4(24) As Variant

```

```

Dim uvol_PCI1(24) As Variant
Dim uvol_PCI2(24) As Variant
Dim uvol_PCI3(24) As Variant
Dim uvol_PCI4(24) As Variant

```

```

Dim uvol_BCF1(24) As Variant
Dim uvol_BCF2(24) As Variant
Dim uvol_BCF3(24) As Variant
Dim uvol_BCF4(24) As Variant

```

```

Dim uvol_PCF1(24) As Variant
Dim uvol_PCF2(24) As Variant
Dim uvol_PCF3(24) As Variant
Dim uvol_PCF4(24) As Variant

```

'Intersection Capacity

```

If VC_1_BC > 0 Then
  Cap_BC = Tot_vol_1_BC / VC_1_BC
Else
  Cap_BC = 0
End If

```

```

If VC_1_PC > 0 Then
  Cap_PC = Tot_vol_1_PC / VC_1_PC
Else

```

```

    Cap_PC = 0
End If

'Initialize excess variables for peak spreading distribution
Excess_BCI = 0
Excess_PCI = 0
Excess_BCF = 0
Excess_PCF = 0

'Calculate Daily Hours of Delay
For hr = 1 To 24

    hr_vol_1 = m_rst("Vol_App1_" & hr)
    hr_vol_2 = m_rst("Vol_App2_" & hr)
    hr_vol_3 = m_rst("Vol_App3_" & hr)
    hr_vol_4 = m_rst("Vol_App4_" & hr)

'Unadjusted volumes
'Base Case, Initial Year
    uvol_BCI1(hr - 1) = hr_vol_1 * (1 + ((Init_Year - Data_Year) * Vol_growth_rate))
    uvol_BCI2(hr - 1) = hr_vol_2 * (1 + ((Init_Year - Data_Year) * Vol_growth_rate))
    uvol_BCI3(hr - 1) = hr_vol_3 * (1 + ((Init_Year - Data_Year) * Vol_growth_rate))
    uvol_BCI4(hr - 1) = hr_vol_4 * (1 + ((Init_Year - Data_Year) * Vol_growth_rate))

    uvol_app12_BCI = uvol_BCI1(hr - 1) + uvol_BCI2(hr - 1)
    uvol_app34_BCI = uvol_BCI3(hr - 1) + uvol_BCI4(hr - 1)
    sum_BCI = uvol_app12_BCI + uvol_app34_BCI

    If uvol_app12_BCI > uvol_app34_BCI Then
        app_max_BCI = uvol_app12_BCI
    Else
        app_max_BCI = uvol_app34_BCI
    End If

'Project Case, Initial Year
    uvol_PCI1(hr - 1) = hr_vol_1 * (1 + ((Init_Year - Data_Year) * Vol_growth_rate)) * _
        (1 - App1_red_PC)
    uvol_PCI2(hr - 1) = hr_vol_2 * (1 + ((Init_Year - Data_Year) * Vol_growth_rate)) * _
        (1 - App2_red_PC)
    uvol_PCI3(hr - 1) = hr_vol_3 * (1 + ((Init_Year - Data_Year) * Vol_growth_rate)) * _
        (1 - App3_red_PC)
    uvol_PCI4(hr - 1) = hr_vol_4 * (1 + ((Init_Year - Data_Year) * Vol_growth_rate)) * _
        (1 - App4_red_PC)

    uvol_app12_PCI = uvol_PCI1(hr - 1) + uvol_PCI2(hr - 1)
    uvol_app34_PCI = uvol_PCI3(hr - 1) + uvol_PCI4(hr - 1)
    sum_PCI = uvol_app12_PCI + uvol_app34_PCI

    If uvol_app12_PCI > uvol_app34_PCI Then
        app_max_PCI = uvol_app12_PCI
    Else
        app_max_PCI = uvol_app34_PCI
    End If

'Base Case, Forecast Year
    uvol_BCF1(hr - 1) = hr_vol_1 * (1 + ((Init_Year - Data_Year + Forecast_Period) * _

```

```

Vol_growth_rate))
uvol_BCF2(hr - 1) = hr_vol_2 * (1 + ((Init_Year - Data_Year + Forecast_Period) * _
Vol_growth_rate))
uvol_BCF3(hr - 1) = hr_vol_3 * (1 + ((Init_Year - Data_Year + Forecast_Period) * _
Vol_growth_rate))
uvol_BCF4(hr - 1) = hr_vol_4 * (1 + ((Init_Year - Data_Year + Forecast_Period) * _
Vol_growth_rate))

```

```

uvol_app12_BCF = uvol_BCF1(hr - 1) + uvol_BCF2(hr - 1)
uvol_app34_BCF = uvol_BCF3(hr - 1) + uvol_BCF4(hr - 1)
sum_BCF = uvol_app12_BCF + uvol_app34_BCF

```

```

If uvol_app12_BCF > uvol_app34_BCF Then
  app_max_BCF = uvol_app12_BCF
Else
  app_max_BCF = uvol_app34_BCF
End If

```

'Project Case, Forecast Year

```

uvol_PCF1(hr - 1) = hr_vol_1 * (1 + ((Init_Year - Data_Year + Forecast_Period) * _
Vol_growth_rate)) * (1 - App1_red_PC)
uvol_PCF2(hr - 1) = hr_vol_2 * (1 + ((Init_Year - Data_Year + Forecast_Period) * _
Vol_growth_rate)) * (1 - App2_red_PC)
uvol_PCF3(hr - 1) = hr_vol_3 * (1 + ((Init_Year - Data_Year + Forecast_Period) * _
Vol_growth_rate)) * (1 - App3_red_PC)
uvol_PCF4(hr - 1) = hr_vol_4 * (1 + ((Init_Year - Data_Year + Forecast_Period) * _
Vol_growth_rate)) * (1 - App4_red_PC)

```

```

uvol_app12_PCF = uvol_PCF1(hr - 1) + uvol_PCF2(hr - 1)
uvol_app34_PCF = uvol_PCF3(hr - 1) + uvol_PCF4(hr - 1)
sum_PCF = uvol_app12_PCF + uvol_app34_PCF

```

```

If uvol_app12_PCF > uvol_app34_PCF Then
  app_max_PCF = uvol_app12_PCF
Else
  app_max_PCF = uvol_app34_PCF
End If

```

'Delay calculations

'Base Case, Initial Year

```

tVol_BCI = uvol_BCI1(hr - 1) + Excess_BCI
If tVol_BCI > Cap_BC Then
  TV_adj_BCI = Cap_BC
  Excess_BCI = TV_adj_BCI - Cap_BC
Else
  TV_adj_BCI = tVol_BC
  Excess_BCI = 0
End If

```

```

MSV_adj_BCI = app_max_BCI * TV_adj_BCI / sum_BCI

```

```

VC_u_BCI = 0.00185 * (TV_adj_BCI / Lanes_BC) - 0.542 * (MSV_adj_BCI / TV_adj_BCI) +
0.918 * VC_1_BC - 0.00147 * (Tot_vol_1_BC / Lanes_BC)

```

```

If VC_u_BCI > 1 Then

```

```

    VC_c_BCI = 1
  Else
    VC_c_BCI = VC_c_BCI
  End If

  D_BCI = (Exp(7.11 + 4.25 * VC_c_BCI + 0.000494 * TV_adj_BCI - 1.319 * _
    (MSV_adj_BCI / TV_adj_BCI) + 0.000000643 * Delay_BC * Tot_vol_1_BC)) / 3600

  Day_Dh_BCI = Day_Dh_BCI + D_BCI

'Project Case, Initial Year
tVol_PCI = uvol_PCI1(hr - 1) + Excess_PCI
If tVol_PCI > Cap_PC Then
  TV_adj_PCI = Cap_PC
  Excess_PCI = TV_adj_PCI - Cap_PC
Else
  TV_adj_PCI = tVol_PC
  Excess_PCI = 0
End If

MSV_adj_PCI = app_max_PCI * TV_adj_PCI / sum_PCI

VC_u_PCI = 0.00185 * (TV_adj_PCI / Lanes_PC) - 0.542 * (MSV_adj_PCI / TV_adj_PCI) +
- 0.918 * VC_1_PC - 0.00147 * (Tot_vol_1_PC / Lanes_PC)

If VC_u_PCI > 1 Then
  VC_c_PCI = 1
Else
  VC_c_PCI = VC_c_PCI
End If

D_PCI = (Exp(7.11 + 4.25 * VC_c_PCI + 0.000494 * TV_adj_PCI - 1.319 * _
  (MSV_adj_PCI / TV_adj_PCI) + 0.000000643 * Delay_PC * Tot_vol_1_PC)) / 3600

Day_Dh_PCI = Day_Dh_PCI + D_PCI

'Base Case, Forecast Year
tVol_BCF = uvol_BCF1(hr - 1) + Excess_BCF
If tVol_BCF > Cap_BC Then
  TV_adj_BCF = Cap_BC
  Excess_BCF = TV_adj_BCF - Cap_BC
Else
  TV_adj_BCF = tVol_BC
  Excess_BCF = 0
End If

MSV_adj_BCF = app_max_BCF * TV_adj_BCF / sum_BCF

VC_u_BCF = 0.00185 * (TV_adj_BCF / Lanes_BC) - 0.542 * (MSV_adj_BCF /
TV_adj_BCF) + _
0.918 * VC_1_BC - 0.00147 * (Tot_vol_1_BC / Lanes_BC)

If VC_u_BCF > 1 Then
  VC_c_BCF = 1
Else

```

```

VC_c_BCF = VC_c_BCF
End If

D_BCF = (Exp(7.11 + 4.25 * VC_c_BCF + 0.000494 * TV_adj_BCF - 1.319 * _
(MSV_adj_BCF / TV_adj_BCF) + 0.000000643 * Delay_BC * Tot_vol_1_BC)) / 3600

Day_Dh_BCF = Day_Dh_BCF + D_BCF

'Project Case, Initial Year
tVol_PCF = uvol_PCF1(hr - 1) + Excess_PCF
If tVol_PCF > Cap_PC Then
  TV_adj_PCF = Cap_PC
  Excess_PCF = TV_adj_PCF - Cap_PC
Else
  TV_adj_PCF = tVol_PC
  Excess_PCF = 0
End If

MSV_adj_PCF = app_max_PCF * TV_adj_PCF / sum_PCF

VC_u_PCF = 0.00185 * (TV_adj_PCF / Lanes_PC) - 0.542 * (MSV_adj_PCF /
TV_adj_PCF) + _
0.918 * VC_1_PC - 0.00147 * (Tot_vol_1_PC / Lanes_PC)

If VC_u_PCF > 1 Then
  VC_c_PCF = 1
Else
  VC_c_PCF = VC_c_PCF
End If

D_PCF = (Exp(7.11 + 4.25 * VC_c_PCF + 0.000494 * TV_adj_PCF - 1.319 * _
(MSV_adj_PCF / TV_adj_PCF) + 0.000000643 * Delay_PC * Tot_vol_1_PC)) / 3600

Day_Dh_PCF = Day_Dh_PCF + D_PCF

Next hr

'Total Travel Time Calculations
Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")

TT_Sav_i = (Day_Dh_BCI - Day_Dh_PCI) * Ann_Daily_Benefit
TT_Sav_f = (Day_Dh_BCF - Day_Dh_PCF) * Ann_Daily_Benefit

'Allot travel time savings between truck and auto
Truck_per = m_rst("Truck_per")
ttsav_truck_i = Truck_per * TT_Sav_i
ttsav_auto_i = (1 - Truck_per) * TT_Sav_i
ttsav_truck_f = Truck_per * TT_Sav_f
ttsav_auto_f = (1 - Truck_per) * TT_Sav_f

'Total travel time benefit - initial and final year
Time_Value_Truck = a_rst("Time_Value_Truck")
Time_Value_Pers = a_rst("Time_Value_Pers")
tt_ben_i = (ttsav_truck_i * Time_Value_Truck) + _
(ttsav_auto_i * AVO_auto * Time_Value_Pers)
tt_ben_f = (ttsav_truck_f * Time_Value_Truck) + _

```

```

      (ttsav_auto_f * AVO_auto * Time_Value_Pers)

'Total Travel Time Calculations
  m_TT_Sav = (TT_Sav_i + TT_Sav_f) * Forecast_Period / 2
  m_TT_Min = m_TT_Sav * 60

  G_tt_ben = (tt_ben_f - tt_ben_i) / (Forecast_Period - 1)
  m_TT_Ben = (tt_ben_i * PofA) + (G_tt_ben * PofG)

'Total operating cost savings - initial and final year

  op_cost_i = (ttsav_truck_i * OpCost_Truck) + _
              (ttsav_auto_i * OpCost_Auto)
  op_cost_f = (ttsav_truck_f * OpCost_Truck) + _
              (ttsav_auto_f * OpCost_Auto)

'User Benefit Calculations
  G_Opcost_Ben = (op_cost_f - op_cost_i) / (Forecast_Period - 1)
  m_User_Ben = (op_cost_i * PofA) + (G_Opcost_Ben * PofG)

'Set module level values (to be displayed)
  m_Values.Add m_TT_Sav, "TT_SAV"
  m_Values.Add m_TT_Min, "TT_MIN"
  m_Values.Add m_TT_Ben, "TT_BEN"
  m_Values.Add m_User_Ben, "USER_BEN"

'Air Pollution Calculations - for diverted auto trips -
'none in current procedure, but keep as placeholder
  m_CO_Tons = 0
  m_VOC_Tons = 0
  m_NOX_Tons = 0
  m_PM10_Tons = 0
  m_Env_Ben = 0

'Set module level values (to be displayed)
  m_Values.Add m_CO_Tons, "CO_TONS"
  m_Values.Add m_VOC_Tons, "VOC_TONS"
  m_Values.Add m_NOX_Tons, "NOX_TONS"
  m_Values.Add m_PM10_Tons, "PM10_TONS"
  m_Values.Add m_Env_Ben, "ENV_BEN"

'Safety Calculations
  Fatality_Cost = a_rst("Fatality_Cost")
  Disable_Cost = a_rst("Disable_Cost")
  Evident_Cost = a_rst("Evident_Cost")
  Possible_Cost = a_rst("Possible_Cost")
  PDO_Cost = a_rst("PDO_Cost")

'Accident reduction factors
  R = 0
  RP = 0

  Dim whereSQL As String
  whereSQL = ReturnSafetyWhereSQL(m_rst)
  If whereSQL <> "" Then
    'Get 1-d array with values or zero for 5 improvement types

```

```

Rarr = ReturnSafetyArray(whereSQL, "Fat_Inj_Red")
RParr = ReturnSafetyArray(whereSQL, "PDO_Red")

If IsArray(Rarr) Then
  For i = 0 To 4
    If i <> 0 Then
      'For fatality and injury accidents
      Rterm = Rterm * ((100 - Rarr(i - 1)) / 100)
      R = R + (Rarr(i) * Rterm)
      'For property damage accidents
      RPterm = RPterm * ((100 - RParr(i - 1)) / 100)
      RP = RP + (RParr(i) * Rterm)
    Else
      'For fatality and injury accidents
      R = Rarr(i)
      Rterm = 1
      'For property damage accidents
      RP = RParr(i)
      RPterm = 1
    End If
  Next
End If
End If

'Accident reduction due to proposed project
ann_fatality = m_rst("3yr_Fatality") * (R / 100) / 3
ann_disable = m_rst("3yr_Disable") * (R / 100) / 3
ann_evident = m_rst("3yr_Evident") * (R / 100) / 3
ann_possible = m_rst("3yr_Possible") * (R / 100) / 3
ann_property = m_rst("3yr_Property") * (RP / 100) / 3

m_Fatality = (-1) * ann_fatality * 20
m_Injury = (-1) * (ann_disable + ann_evident + ann_possible) * 20
m_Property = (-1) * ann_property * 20

'Calculate safety benefits
m_Safety_Ben = ((ann_fatality * Fatality_Cost) +
(ann_disable * Disable_Cost) + (ann_evident * Evident_Cost) +
(ann_possible * Possible_Cost) + (ann_property * PDO_Cost)) * PofA

'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
'Capital Cost
For i = 1 To 5
  m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
  1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
  1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
  1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _

```



```

    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
    m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations

```

```

    Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
    If (m_Total_Cost - m_Terminal_Cost) = 0 Then
        m_BCR = 0
    Else
        m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
    End If
    If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
        m_WSDOT_BCR = 0
    Else
        m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
    End If
    'Set module level values (to be displayed)
    m_Values.Add m_Total_Cost, "TOTAL_COST"
    m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
    m_Values.Add m_Federal_Cost, "FEDERAL_COST"
    m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
    m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
    m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
    m_Values.Add m_Other1_Cost, "OTHER1_COST"
    m_Values.Add m_Other2_Cost, "OTHER2_COST"
    m_Values.Add m_Other3_Cost, "OTHER3_COST"
    m_Values.Add m_Cap_Cost, "CAP_COST"
    m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
    m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
    m_Values.Add m_Other1_Cap, "OTHER1_CAP"
    m_Values.Add m_Other2_Cap, "OTHER2_CAP"
    m_Values.Add m_Other3_Cap, "OTHER3_CAP"
    m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
    m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
    m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
    m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
    m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
    m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
    m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
    m_Values.Add m_WSDOT_OM, "WSDOT_OM"
    m_Values.Add m_Federal_OM, "FEDERAL_OM"
    m_Values.Add m_Other1_OM, "OTHER1_OM"
    m_Values.Add m_Other2_OM, "OTHER2_OM"
    m_Values.Add m_Other3_OM, "OTHER3_OM"
    m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
    m_Values.Add m_BCR, "BCR"
    m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next
    'Calculation for the System Operation and Maintenance
    m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
    m_Values.Add m_Sys_OM, "SYS_OM"
    'Calculation for the System Preservation
    m_Sys_Pres = 100 * m_rst("Q2A")
    m_Values.Add m_Sys_Pres, "SYS_PRES"
    'Calculation for the Special Needs Transportation

```

```

    m_Sp_Needs = 100 * m_rst("Q3A")
    m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
  Else
    m_Cong_Rel = (m_rst("Q4") * 50)
  End If
  m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
  Else
    m_Safety = m_rst("Q7B") * 50
  End If
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  'The -1 Multiplication Corrects the negative sign introduced for true
  m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
  Else
    m_Collab = (m_rst("Q10A") * 50) + 50
  End If
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
  If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + m_rst("Q14A") * 50
  Else
    m_Air_Qual = 0
  End If
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _

```

```
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub
```

Park and Ride

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim a_rst As DAO.Recordset

Dim m_qryName As String

Dim m_calcsComplete As Boolean

Dim m_Values As ValueCollection

'Variables used in intermediate calculations (variants)

Dim m_TT_Sav

Dim m_OpCost

Dim m_Approach

Dim m_WSDOT_Cap

Dim m_Federal_Cap

Dim m_Other1_Cap

Dim m_Other2_Cap

Dim m_Other3_Cap

Dim m_WSDOT_OM

Dim m_Federal_OM

Dim m_Other1_OM

Dim m_Other2_OM

Dim m_Other3_OM

Dim m_FishBarrier_Ben

Dim m_StormWater_Ben

Dim m_NoiseBarrier_Ben

Dim m_FishBarrier_Cap

Dim m_StormWater_Cap

Dim m_NoiseBarrier_Cap

'Variables to hold the calculated values (variants)

Dim m_Total_Benefit

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_EnvRetrofit_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Other1_Cost

Dim m_Other2_Cost

Dim m_Other3_Cost

Dim m_Cap_Cost

```
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
Private Sub Class_Initialize()
    m_qryName = "hwy_calc_Park_Ride"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
```

```
    Set m_rst = qryDef.OpenRecordset
```

```
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If
```

```

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
    Forecast_Period = 20

'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period) - (Forecast_Period / _
        ((1 + Discount_Rate) ^ Forecast_Period)))
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)
    Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")

'Determine operating cost values
    Auto_OpCostHr_Full = a_rst("Veh_OpCostHr_Full")
    Auto_OpCostHr_Direct = a_rst("Veh_OpCostHr_Direct")
    If a_rst("Full_Cost") Then
        m_Approach = "Full cost"
        OpCost_Auto = Auto_OpCostHr_Full
    Else

```

```

    m_Approach = "Direct cost"
    OpCost_Auto = Auto_OpCostHr_Direct
End If

'General Data (applies to all destinations)
park_space = m_rst("park_space")
capacity_year = m_rst("capacity_year")
walkbike_20 = m_rst("walkbike_20")
percent_used_1 = m_rst("percent_used_1")
percent_used_20 = m_rst("percent_used_20")

'NEED TO CALCULATE FOR EACH OF UP TO FIVE DESTINATIONS
'Begin to loop through each item in the recordset
While Not m_rst.EOF

    Percent_Dest = m_rst("percent_dest")
    Dist = m_rst("Dist")
    HOV_length = m_rst("HOV_length")
    HOV_speed = m_rst("HOV_speed")
    headway = m_rst("headway")
    transit_tt = m_rst("transit_tt")
    expr_pass_cost = m_rst("expr_pass_cost")
    loc_pass_cost = m_rst("loc_pass_cost")
    AVO_carpool = m_rst("avo_carpool")
    carpool_wait = m_rst("carpool_wait")
    park_cost = m_rst("park_cost")

    Dest_vol_i = park_space * AVO_to_lot * percent_used_1 * walkbike_20

    'Travel Time Savings
    'New transit riders
    newtrans_trav_i = (Dist / GP_speed - HOV_length / HOV_speed - _
        (Dist - HOV_length) / GP_speed) * 60

    If headway <= 10 Then
        newtrans_wait_i = headway / 2
    Else
        newtrans_wait_i = (headway) ^ (0.5)
    End If

    newtrans_tt_i = newtrans_trav_i - newtrans_wait_i

    'Existing transit riders
    oldtrans_trav_i = (Dist / GP_speed - HOV_length / HOV_speed - _
        (Dist - HOV_length) / GP_speed) * 60
    oldtrans_wait_i = 0
    oldtrans_tt_i = oldtrans_trav_i - oldtrans_wait_i

    'New carpoolers
    newcarpl_trav_i = (Dist / GP_speed - HOV_length / HOV_speed - _
        (Dist - HOV_length) / GP_speed) * 60
    newcarpl_wait_i = carpool_wait
    newcarpl_tt_i = newcarpl_trav_i - newcarpl_wait_i

    'Total travel time savings

```



```

TT_destMin_i = Ann_Daily_Benefit * Dest_vol_i * (newtransit * newtrans_tt_i + _
oldtransit * oldtrans_tt_i + newcarpool * newcarpl_tt_i)

TT_destMin_f = TT_destSav_i * (percent_used_20 / percent_used_1)

TT_Min_i = TT_Min_i + TT_destMin_i
TT_Min_f = TT_Min_f + TT_destMin_f

'User Savings
'New transit riders
newtrans_user_i = (12 * (park_cost - expr_pass_cost)) + _
(Ann_Daily_Benefits * Dist * OpCost_Auto)

'Existing transit riders
oldtrans_user_i = 12 * (loc_pass_cost - expr_pass_cost)

'New carpoolers
If AVO_carpool > 0 Then
newcarpl_user_i = (12 * park_cost + (Ann_Daily_Benefits * OpCost_Auto * Dist)) / _
AVO_carpool
Else
newcarpl_user_i = 0
End If

'Total travel time savings
User_destBen_i = Dest_vol_i * (newtransit * newtrans_user_i + _
oldtransit * oldtrans_user_i + newcarpool * newcarpl_user_i)

User_destBen_f = User_destBen_i * (percent_used_20 / percent_used_1)

User_Ben_i = User_Ben_i + User_destBen_i
User_Ben_f = User_Ben_f + User_destBen_f

'NEXT DESTINATION
m_rst.MoveNext
Wend

'Reset records so we're back at the beginning - this method can be removed if
'm_rst is not accessed again in code below.
m_rst.MoveFirst

'Total travel time benefit - initial and final year
TT_Sav_i = TT_Min_i / 60
TT_Sav_f = TT_Min_f / 60
Time_Value_Pers = a_rst("Time_Value_Pers")
tt_ben_i = (TT_Sav_i * Time_Value_Pers)
tt_ben_f = (TT_Sav_f * Time_Value_Pers)
G_tt_ben = (tt_ben_f - tt_ben_i) / (Forecast_Period - 1)
m_TT_Ben = (tt_ben_i * PofA) + (G_tt_ben * PofG)

'Total Travel Time Calculations
m_TT_Sav = (TT_Sav_i + TT_Sav_f) * Forecast_Period / 2

'User Benefit Calculations
G_User_Ben = (User_Ben_f - User_Ben_i) / (Forecast_Period - 1)

```

```

m_User_Ben = (User_Ben_i * PofA) + (G_User_Ben * PofG)

'Set module level values (to be displayed)
m_Values.Add m_TT_Sav, "TT_SAV"
m_Values.Add m_TT_Min, "TT_MIN"
m_Values.Add m_TT_Ben, "TT_BEN"
m_Values.Add m_User_Ben, "USER_BEN"

'Air Pollution Calculations - for diverted auto trips -
'none in current procedure, but keep as placeholder
m_CO_Tons = 0
m_VOC_Tons = 0
m_NOX_Tons = 0
m_PM10_Tons = 0
m_Env_Ben = 0

'Set module level values (to be displayed)
m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"
m_Values.Add m_PM10_Tons, "PM10_TONS"
m_Values.Add m_Env_Ben, "ENV_BEN"

'Safety Calculations
Fatality_Cost = a_rst("Fatality_Cost")
Disable_Cost = a_rst("Disable_Cost")
Evident_Cost = a_rst("Evident_Cost")
Possible_Cost = a_rst("Possible_Cost")
PDO_Cost = a_rst("PDO_Cost")

'Accident reduction factors
R = 0
RP = 0

Dim whereSQL As String
whereSQL = ReturnSafetyWhereSQL(m_rst)
If whereSQL <> "" Then
    'Get 1-d array with values or zero for 5 improvement types
    Rarr = ReturnSafetyArray(whereSQL, "Fat_Inj_Red")
    RParr = ReturnSafetyArray(whereSQL, "PDO_Red")

    If IsArray(Rarr) Then
        For i = 0 To 4
            If i <> 0 Then
                'For fatality and injury accidents
                Rterm = Rterm * ((100 - Rarr(i - 1)) / 100)
                R = R + (Rarr(i) * Rterm)
                'For property damage accidents
                RPterm = RPterm * ((100 - RParr(i - 1)) / 100)
                RP = RP + (RParr(i) * Rterm)
            Else
                'For fatality and injury accidents
                R = Rarr(i)
                Rterm = 1
                'For property damage accidents

```

```

        RP = Rarr(i)
        RPterm = 1
    End If
Next
End If
End If

'Accident reduction due to proposed project
ann_fatality = m_rst("3yr_Fatality") * (R / 100) / 3
ann_disable = m_rst("3yr_Disable") * (R / 100) / 3
ann_evident = m_rst("3yr_Evident") * (R / 100) / 3
ann_possible = m_rst("3yr_Possible") * (R / 100) / 3
ann_property = m_rst("3yr_Property") * (RP / 100) / 3

m_Fatality = (-1) * ann_fatality * 20
m_Injury = (-1) * (ann_disable + ann_evident + ann_possible) * 20
m_Property = (-1) * ann_property * 20

'Calculate safety benefits
m_Safety_Ben = ((ann_fatality * Fatality_Cost) +
(ann_disable * Disable_Cost) + (ann_evident * Evident_Cost) +
(ann_possible * Possible_Cost) + (ann_property * PDO_Cost)) * PofA

'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
'Capital Cost
For i = 1 To 5
    m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) *
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) *
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) *
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) *
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) *
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost

```

```

m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
If (m_Total_Cost - m_Terminal_Cost) = 0 Then
m_BCR = 0
Else
m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
End If
If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
m_WSDOT_BCR = 0
Else
m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
End If
'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"

```

```

m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

```

'Calculation for the System Operation and Maintenance
m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
m_Sys_Pres = 100 * m_rst("Q2A")
m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
m_Sp_Needs = 100 * m_rst("Q3A")
m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
Else
    m_Cong_Rel = (m_rst("Q4") * 50)
End If
m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)

```

```

Else
    m_Safety = m_rst("Q7B") * 50
End If
m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
m_Security = 100 * m_rst("Q8A")
m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
'The -1 Multiplication Corrects the negative sign introduced for true
m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
Else
    m_Collab = (m_rst("Q10A") * 50) + 50
End If
m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + m_rst("Q14A") * 50
Else
    m_Air_Qual = 0
End If
m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
    (33 * m_rst("Q15C"))
m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
    (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
    (1 + m_rst("Q16E") + m_rst("Q16F"))
m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
m_Resource = 100 * m_rst("Q17")
m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

```

```
    GetItemValue = rtnval  
End Function
```

```
Private Sub Class_Terminate()  
    a_rst.Close  
    m_rst.Close  
    m_dbs.Close  
    Set a_rst = Nothing  
    Set m_rst = Nothing  
    Set m_dbs = Nothing  
    Set m_Values = Nothing  
End Sub
```

Two-Way-Left-Turn Lane

Option Compare Database

'Variables for the class level database objects

```
Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection
```

'Variables used in intermediate calculations (variants)

```
Dim m_TT_Sav
Dim m_OpCost
Dim m_Approach
Dim m_Delay_Method
Dim m_WSDOT_Cap
Dim m_Federal_Cap
Dim m_Other1_Cap
Dim m_Other2_Cap
Dim m_Other3_Cap
Dim m_WSDOT_OM
Dim m_Federal_OM
Dim m_Other1_OM
Dim m_Other2_OM
Dim m_Other3_OM
Dim m_FishBarrier_Ben
Dim m_StormWater_Ben
Dim m_NoiseBarrier_Ben
Dim m_FishBarrier_Cap
Dim m_StormWater_Cap
Dim m_NoiseBarrier_Cap
```

'Variables to hold the calculated values (variants)

```
Dim m_Total_Benefit
Dim m_TT_Min
Dim m_TT_Ben
Dim m_User_Ben
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_EnvRetrofit_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
Dim m_Other2_Cost
```



```
Dim m_Other3_Cost
Dim m_Cap_Cost
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

'Variables to hold the outcome objective scores (variants)

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
Private Sub Class_Initialize()
    m_qryName = "hwy_calc_TWLTL"
    m_calcsComplete = False
End Sub
```

Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean

```
Dim atype As String
Dim qryDef As DAO.QueryDef
```

```
Set m_dbs = CurrentDb
```

```
' Open QueryDef object with one parameters.
Set qryDef = m_dbs.QueryDefs(m_qryName)
qryDef.Parameters!projID = pid
```

```
' Set recordset to new values.
Set m_rst = qryDef.OpenRecordset
' If data is entered for this particular project then calculate all scores
If Not m_rst.EOF Then
```

```
    'Initialize new values collection to store calculated values
    Set m_Values = New ValueCollection
```

```
    'If you are running MICA use the scenario assumptions
    If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
        atype = "prj_Project_Assumptions"
    Else
        atype = "prj_Global_Assumptions"
        pid = asmptnID
```

```

End If

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
    Forecast_Period = 20

'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / _
        ((1 + Discount_Rate) ^ Forecast_Period)))
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Calculate delay reduction - exit if no curve was selected
    ADT_i = m_rst("ADT_i")
    ADT_f = m_rst("ADT_f")
    m_Length = m_rst("RoadLength")
    AVO_auto_pk = m_rst("AVO_auto_pk")
    AVO_auto_npk = m_rst("AVO_auto_npk")

```

```
'Determine operating cost values
Auto_OpCostHr_Full = a_rst("Veh_OpCostHr_Full")
Auto_OpCostHr_Direct = a_rst("Veh_OpCostHr_Direct")
Truck_OpCostHr_Full = a_rst("Truck_OpCostHr_Full")
Truck_OpCostHr_Direct = a_rst("Truck_OpCostHr_Direct")
  If a_rst("Full_Cost") Then
    m_Approach = "Full cost"
    OpCost_Auto = Auto_OpCostHr_Full
    OpCost_Truck = Truck_OpCostHr_Full
  Else
    m_Approach = "Direct cost"
    OpCost_Auto = Auto_OpCostHr_Direct
    OpCost_Truck = Truck_OpCostHr_Direct
  End If
```

```
Dim crvid As Integer
```

```
If Not IsNumeric(m_rst("Curve_ID")) Then Exit Sub
crvid = m_rst("Curve_ID")
'these are the arrays that hold all values to be used later
Dim Dir1 As Variant
Dim Dir2 As Variant
```

```
Dim NW_per(24) As Variant
Dim NWpki_per(24) As Variant
Dim NWpkf_per(24) As Variant
Dim SE_per(24) As Variant
Dim SEpki_per(24) As Variant
Dim SEpkf_per(24) As Variant
Dim Both_per(24) As Variant
```

```
'lkup function for NCHRP values
Dim lkup_nchrp As New lkup_TWLTL_coef
Dim Median_ID As Integer
```

```
'Calculate AM peak hour percentages
```

```
AMpk_beg_i = m_rst("AMpk_beg_i")
AMpk_end_i = m_rst("AMpk_end_i")
AMpk_beg_f = m_rst("AMpk_beg_f")
AMpk_end_f = m_rst("AMpk_end_f")
```

```
AM_Max_i = 0
AM_Max_f = 0
```

```
For hr = 1 To 12
  NW_per(hr - 1) = Dir1(hr - 1)
  SE_per(hr - 1) = Dir2(hr - 1)
  Both_per(hr - 1) = NW_per(hr - 1) + SE_per(hr - 1)

  If hr < AMpeak_begin_i Then
    NWpki_per(hr - 1) = 0
    SEpki_per(hr - 1) = 0
  Else
    If hr > AMpeak_end_i Then
```

```

        NWpki_per(hr - 1) = 0
        SEpki_per(hr - 1) = 0
    Else
        NWpki_per(hr - 1) = NW_per(hr - 1)
        SEpki_per(hr - 1) = SE_per(hr - 1)
    End If
End If

peak_per_day_i = peak_per_day_i + NWpki_per(hr - 1) + SEpki_per(hr - 1)

If AM_Max_i < Both_per(hr - 1) Then
    AM_Max_i = Both_per(hr - 1)
End If

If hr < AMpeak_begin_f Then
    NWpkf_per(hr - 1) = 0
    SEpkf_per(hr - 1) = 0
Else
    If hr > AMpeak_end_i Then
        NWpkf_per(hr - 1) = 0
        SEpkf_per(hr - 1) = 0
    Else
        NWpkf_per(hr - 1) = NW_per(hr - 1)
        SEpkf_per(hr - 1) = SE_per(hr - 1)
    End If
End If

peak_per_day_f = peak_per_day_f + NWpkf_per(hr - 1) + SEpkf_per(hr - 1)

If AM_Max_f < Both_per(hr - 1) Then
    AM_Max_f = Both_per(hr - 1)
End If

Next hr

'Calculate PM peak hour percentages

PMpk_beg_i = m_rst("PMpk_beg_i")
PMpk_end_i = m_rst("PMpk_end_i")
PMpk_beg_f = m_rst("PMpk_beg_f")
PMpk_end_f = m_rst("PMpk_end_f")

PM_Max_i = 0
PM_Max_f = 0

For hr = 13 To 24
    NW_per(hr - 1) = Dir1(hr - 1)
    SE_per(hr - 1) = Dir2(hr - 1)
    Both_per(hr - 1) = NW_per(hr - 1) + SE_per(hr - 1)

    If hr < PMpeak_begin_i Then
        NWpki_per(hr - 1) = 0
        SEpki_per(hr - 1) = 0
    Else
        If hr > PMpeak_end_i Then
            NWpki_per(hr - 1) = 0

```

```

        SEpki_per(hr - 1) = 0
    Else
        NWpki_per(hr - 1) = NW_per(hr - 1)
        SEpki_per(hr - 1) = SE_per(hr - 1)
    End If
End If

peak_per_day_i = peak_per_day_i + NWpki_per(hr - 1) + SEpki_per(hr - 1)

If PM_Max_i < Both_per(hr - 1) Then
    PM_Max_i = Both_per(hr - 1)
End If

If hr < PMpeak_begin_f Then
    NWpkf_per(hr - 1) = 0
    SEpkf_per(hr - 1) = 0
Else
    If hr > PMpeak_end_i Then
        NWpkf_per(hr - 1) = 0
        SEpkf_per(hr - 1) = 0
    Else
        NWpkf_per(hr - 1) = NW_per(hr - 1)
        SEpkf_per(hr - 1) = SE_per(hr - 1)
    End If
End If

peak_per_day_f = peak_per_day_f + NWpkf_per(hr - 1) + SEpkf_per(hr - 1)

If PM_Max_f < Both_per(hr - 1) Then
    PM_Max_f = Both_per(hr - 1)
End If

Next hr

'Calculate the Daily / Peak Hour Benefit Ratio

If AM_Max_i > PM_Max_i Then
    Day_Max_i = AM_Max_i
Else
    Day_Max_i = PM_Max_i
End If

If AM_Max_f > PM_Max_f Then
    Day_Max_f = AM_Max_f
Else
    Day_Max_f = PM_Max_f
End If

Day_PkHr_Rat_i = Day_Max_i / peak_per_day_i
Day_PkHr_Rat_f = Day_Max_f / peak_per_day_f

'Calculate travel time savings

If m_rst("Nonpeak_Lns_PC") <= 1 Then

    'Harwood/St. John Delay Calculation Method

```

```

    m_Delay_Method = "Harwood/St. John"

'Set Facility Characteristics
Access_Space = m_rst("Access_Space_PC")

For i = 3 To 4
'Peak Direction
    de_pk = -6.87 + 0.058 * (m_rst("Thr_npk_vol_" & i) + m_rst("Rt_npk_vol_" & i))
    If Access_Space = 0 Then
        a_pk = 0
    Else
        a_pk = 5280 / Access_Space
    End If
    D_pk = m_rst("Lt_pk_vol_" & i) * a_pk * m_Length * de_pk / 3600

'Nonpeak Direction
    de_npk = -6.87 + 0.058 * (m_rst("Thr_pk_vol_" & i) + m_rst("Rt_pk_vol_" & i))
    a_npk = a_pk
    D_npk = m_rst("Lt_npk_vol_" & i) * a_npk * m_Length * de_npk / 3600

TTS = D_pk + D_npk

If i = 3 Then
    TTS_i = TTS
Else
    TTS_f = TTS
End If

Next i

Else

'NCHRP Method
    m_Delay_Method = "NCHRP 395"

For i = 1 To 4

'Set Facility Characteristics

If i <= 2 Then
    Peak_Lns = m_rst("Peak_Lns_BC")
    Nonpeak_Lns = m_rst("Nonpeak_Lns_BC")
    Median = m_rst("Median_BC")
    Access_Space = m_rst("Access_Space_BC")
    Access_Control = m_rst("Access_Control_BC")
    Median_ID = m_rst("Median_BC")
Else
    Peak_Lns = m_rst("Peak_Lns_PC")
    Nonpeak_Lns = m_rst("Nonpeak_Lns_PC")
    Median = m_rst("Median_PC")
    Access_Space = m_rst("Access_Space_PC")
    Access_Control = m_rst("Access_Control_PC")
    Median_ID = m_rst("Median_PC")
End If

'Set Coefficient Values

```

```

With lkup_nchrp
  b0_th = .RetrieveRate(Median_ID, c_b0_th)
  b1_th = .RetrieveRate(Median_ID, c_b1_th)
  b2_th = .RetrieveRate(Median_ID, c_b2_th)
  ltr_th = .RetrieveRate(Median_ID, c_ltr_th)
  cm_th = .RetrieveRate(Median_ID, c_cm_th)
  b0_lt = .RetrieveRate(Median_ID, c_b0_lt)
  b1_lt = .RetrieveRate(Median_ID, c_b1_lt)
  b2_lt = .RetrieveRate(Median_ID, c_b2_lt)
  b3_lt = .RetrieveRate(Median_ID, c_b3_lt)
End With

'Delay Equations
'Through delay in peak direction
N_pk = Peak_Lns * 2
X_pk = m_rst("Lt_pk_vol_" & i) / (1800 * _
  (1 - ((2 * m_rst("Thr_npk_vol_" & i)) / (N_pk * cm_th))))
If X_pk < 1 Then
  X_pk = X_pk
Else
  X_pk = 1
End If
Y_pk = m_rst("Thr_pk_vol_" & i) / (1800 * ((N_pk / 2) - _
  (X_pk * (1 - ltr_th))))
ft_pk = Y_pk / (1 - Y_pk)
fl_pk = b1_th + ((1 - X_pk) * (X_pk ^ b2_th))
det_pk = b0_th * ft_pk * fl_pk
Na_pk = 5280 / Access_Space * m_Length
Dt_pk = (m_rst("Thr_pk_vol_" & i) + _
  m_rst("Rt_pk_vol_" & i)) * det_pk * Na_pk / 3600

'Through delay in nonpeak direction
N_npk = Peak_Lns * 2
X_npk = m_rst("Lt_npk_vol_" & i) / (1800 * _
  (1 - ((2 * m_rst("Thr_npk_vol_" & i)) / (N_npk * cm_th))))
If X_npk < 1 Then
  X_npk = X_npk
Else
  X_npk = 1
End If
Y_npk = m_rst("Thr_npk_vol_" & i) / (1800 * ((N_npk / 2) - _
  (X_npk * (1 - ltr_th))))
ft_npk = Y_npk / (1 - Y_npk)
fl_npk = b1_th + ((1 - X_npk) * (X_npk ^ b2_th))
det_npk = b0_th * ft_npk * fl_npk
Na_npk = 5280 / Access_Space * m_Length
Dt_npk = (m_rst("Thr_npk_vol_" & i) + _
  m_rst("Rt_npk_vol_" & i)) * det_npk * Na_npk / 3600

'Left turn delay in peak direction
If Peak_Lns = 2 Then
  g_pk = b1_lt
Else
  If Peak_Lns = 3 Then
    g_pk = b3_lt
  Else

```

```

    g_pk = 0
  End If
End If
u_pk = 3600 / (m_rst("Thr_vol_npk_" & i) * _
  (Exp(-m_rst("Thr_vol_npk_" & i) * g_pk / 3600)))
If m_rst("Lt_vol_pk_" & i) = 0 Then
  dla_pk = 0
Else
  dla_pk = b0_lt * u_pk * (1 + ((m_rst("Lt_vol_pk_" & i) * _
    u_pk / 3600) ^ b2_lt))
End If
DI_pk = m_rst("Lt_vol_pk_" & i) * Na_pk * dla_pk / 3600

'Left turn delay in nonpeak direction
If Nonpeak_Lns = 2 Then
  g_npk = b1_lt
Else
  If Nonpeak_Lns = 3 Then
    g_npk = b3_lt
  Else
    g_npk = 0
  End If
End If
u_npk = 3600 / (m_rst("Thr_vol_pk_" & i) * _
  (Exp(-m_rst("Thr_vol_pk_" & i) * g_npk / 3600)))
If m_rst("Lt_vol_npk_" & i) = 0 Then
  dla_npk = 0
Else
  dla_npk = b0_lt * u_npk * (1 + ((m_rst("Lt_vol_npk_" & i) * _
    u_npk / 3600) ^ b2_lt))
End If
DI_npk = m_rst("Lt_vol_npk_" & i) * Na_npk * dla_npk / 3600

If i = 1 Then
  Dt_pk_bci = Dt_pk
  Dt_npk_bci = Dt_npk
  DI_pk_bci = DI_pk
  DI_npk_bci = DI_npk
Else
  If i = 2 Then
    Dt_pk_bcf = Dt_pk
    Dt_npk_bcf = Dt_npk
    DI_pk_bcf = DI_pk
    DI_npk_bcf = DI_npk
  Else
    If i = 3 Then
      Dt_pk_pci = Dt_pk
      Dt_npk_pci = Dt_npk
      DI_pk_pci = DI_pk
      DI_npk_pci = DI_npk
    Else
      Dt_pk_pcf = Dt_pk
      Dt_npk_pcf = Dt_npk
      DI_pk_pcf = DI_pk
      DI_npk_pcf = DI_npk
    End If
  End If
End If

```



```

    End If
  End If

  Next i

  TTS_i = (Dt_pk_bci + Dt_npk_bci + DI_pk_bci + DI_npk_bci) - _
          (Dt_pk_pci + Dt_npk_pci + DI_pk_pci + DI_npk_pci)
  TTS_f = (Dt_pk_bcf + Dt_npk_bcf + DI_pk_bcf + DI_npk_bcf) - _
          (Dt_pk_pcf + Dt_npk_pcf + DI_pk_pcf + DI_npk_pcf)

End If

'Convert tt savings from hourly to daily and allot between truck and auto
Truck_per = m_rst("Truck_per")
Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
Auto_per = 1 - Truck_per
nonpeak_per_day_i = 1 - peak_per_day_i
nonpeak_per_day_f = 1 - peak_per_day_f

ttsav_truck_i = TTS_i * Truck_per * Day_PkHr_Rat_i * Ann_Daily_Benefit
ttsav_auto_i = TTS_i * Auto_per * (AVO_auto_pk * peak_per_day_i + _
  AVO_auto_npk * nonpeak_per_day_i) * Day_PkHr_Rat_i * _
  Ann_Daily_Benefit
ttsav_truck_f = TTS_f * Truck_per * Day_PkHr_Rat_i * Ann_Daily_Benefit
ttsav_auto_f = TTS_f * Auto_per * (AVO_auto_pk * peak_per_day_f + _
  AVO_auto_npk * nonpeak_per_day_f) * Day_PkHr_Rat_i * _
  Ann_Daily_Benefit

'Total travel time benefit - initial and final year
Time_Value_Truck = a_rst("Time_Value_Truck")
Time_Value_Pers = a_rst("Time_Value_Pers")
tt_ben_i = (ttsav_truck_i * Time_Value_Truck) + _
  (ttsav_auto_i * AVO_auto * Time_Value_Pers)
tt_ben_f = (ttsav_truck_f * Time_Value_Truck) + _
  (ttsav_auto_f * AVO_auto * Time_Value_Pers)

'Total Travel Time Calculations
m_TT_Sav = (tt_sav_i + tt_sav_f) * Forecast_Period / 2
m_TT_Min = m_TT_Sav * 60

G_tt_ben = (tt_ben_f - tt_ben_i) / (Forecast_Period - 1)
m_TT_Ben = (tt_ben_i * PofA) + (G_tt_ben * PofG)

'Total operating cost savings - initial and final year

op_cost_i = (ttsav_truck_i * OpCost_Truck) + _
  (ttsav_auto_i * OpCost_Auto)
op_cost_f = (ttsav_truck_f * OpCost_Truck) + _
  (ttsav_auto_f * OpCost_Auto)

'User Benefit Calculations
G_Opcost_Ben = (op_cost_f - op_cost_i) / (Forecast_Period - 1)
m_User_Ben = (op_cost_i * PofA) + (G_Opcost_Ben * PofG)

'Set module level values (to be displayed)
m_Values.Add m_TT_Sav, "TT_SAV"

```

```

m_Values.Add m_TT_Min, "TT_MIN"
m_Values.Add m_TT_Ben, "TT_BEN"
m_Values.Add m_User_Ben, "USER_BEN"

```

```

'Air Pollution Calculations - for diverted auto trips -
'none in current procedure, but keep as placeholder

```

```

m_CO_Tons = 0
m_VOC_Tons = 0
m_NOX_Tons = 0
m_PM10_Tons = 0
m_Env_Ben = 0

```

```

'Set module level values (to be displayed)
m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"
m_Values.Add m_PM10_Tons, "PM10_TONS"
m_Values.Add m_Env_Ben, "ENV_BEN"

```

```

'Safety Calculations

```

```

Fatality_Cost = a_rst("Fatality_Cost")
Disable_Cost = a_rst("Disable_Cost")
Evident_Cost = a_rst("Evident_Cost")
Possible_Cost = a_rst("Possible_Cost")
PDO_Cost = a_rst("PDO_Cost")

```

```

'Accident reduction factors

```

```

R = 0
RP = 0

```

```

Dim whereSQL As String

```

```

whereSQL = ReturnSafetyWhereSQL(m_rst)

```

```

If whereSQL <> "" Then

```

```

'Get 1-d array with values or zero for 5 improvement types

```

```

Rarr = ReturnSafetyArray(whereSQL, "Fat_Inj_Red")
RParr = ReturnSafetyArray(whereSQL, "PDO_Red")

```

```

If IsArray(Rarr) Then

```

```

For i = 0 To 4

```

```

If i <> 0 Then

```

```

'For fatality and injury accidents

```

```

Rterm = Rterm * ((100 - Rarr(i - 1)) / 100)

```

```

R = R + (Rarr(i) * Rterm)

```

```

'For property damage accidents

```

```

RPterm = RPterm * ((100 - RParr(i - 1)) / 100)

```

```

RP = RP + (RParr(i) * RPterm)

```

```

Else

```

```

'For fatality and injury accidents

```

```

R = Rarr(i)

```

```

Rterm = 1

```

```

'For property damage accidents

```

```

RP = RParr(i)

```

```

RPterm = 1

```

```

End If

```

```

Next

```

```

End If

```

```

End If

'Accident reduction due to proposed project
ann_fatality = m_rst("3yr_Fatality") * (R / 100) / 3
ann_disable = m_rst("3yr_Disable") * (R / 100) / 3
ann_evident = m_rst("3yr_Evident") * (R / 100) / 3
ann_possible = m_rst("3yr_Possible") * (R / 100) / 3
ann_property = m_rst("3yr_Property") * (RP / 100) / 3

m_Fatality = (-1) * ann_fatality * 20
m_Injury = (-1) * (ann_disable + ann_evident + ann_possible) * 20
m_Property = (-1) * ann_property * 20

'Calculate safety benefits
m_Safety_Ben = ((ann_fatality * Fatality_Cost) + _
(ann_disable * Disable_Cost) + (ann_evident * Evident_Cost) + _
(ann_possible * Possible_Cost) + (ann_property * PDO_Cost)) * PofA

'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
'Capital Cost
For i = 1 To 5
m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + _
m_Other2_Cap + m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + _
m_Other2_OM + m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM

```

```

m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

```

```

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

```

```

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)

```

```

'Environmental Retrofit Calculations

```

```

fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

```

```

'Environmental Retrofit Costs

```

```

For i = 1 To 5

```

```

    m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))

```

```

Next

```

```

m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

```

```

'Environmental Retrofit Benefits

```

```

m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

```

```

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

```

```

'Benefit-Cost Calculations

```

```

Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + _
m_Safety_Ben + m_EnvRetrofit_Ben

```

```

If (m_Total_Cost - m_Terminal_Cost) = 0 Then

```

```

    m_BCR = 0

```

```

Else

```

```

    m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)

```

```

End If

```

```

If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then

```

```

    m_WSDOT_BCR = 0

```

```

Else

```

```

    m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)

```

```

End If

```

```

'Set module level values (to be displayed)

```

```

m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"

```

```

m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

```

'Calculation for the System Operation and Maintenance
m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
m_Sys_Pres = 100 * m_rst("Q2A")
m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
m_Sp_Needs = 100 * m_rst("Q3A")
m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
Else
    m_Cong_Rel = (m_rst("Q4") * 50)
End If
m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
Else
    m_Safety = m_rst("Q7B") * 50
End If
m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security

```

```

    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
'The -1 Multiplication Corrects the negative sign introduced for true
    m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E")))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") < 5 Then
        m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
    Else
        m_Collab = (m_rst("Q10A") * 50) + 50
    End If
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    If m_rst("Q14B") = 1 Then
        m_Air_Qual = 50 + m_rst("Q14A") * 50
    Else
        m_Air_Qual = 0
    End If
    m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close

```

```
m_rst.Close  
m_dbs.Close  
Set a_rst = Nothing  
Set m_rst = Nothing  
Set m_dbs = Nothing  
Set m_Values = Nothing  
End Sub
```

Program Code for Highway Preservation Projects

Pavement Preservation Projects

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim m_Values As Collection

'Variables used in intermediate calculations (variants)

Dim m_VitalScr
Dim m_NonVitalScr
Dim m_Prob(1 To 5)
Dim m_VMT_PC(1 To 5)

'Variables to hold the calculated values (variants)

Dim m_TT_Min
Dim m_TT_Ben
Dim m_User_Ben
Dim m_User_Transfer
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR

'Variables to hold the outcome objective scores (variants)

Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource

'Speed Assumed for lookup table
Private Const AUTOSPEED = 50

Public Function SetProjectNumber(pid As Long) As Boolean

```

Dim rtnval As Boolean
Dim qryDef As DAO.QueryDef

'Initialize return value to false - set true if one record is returned
rtnval = False

Set m_dbs = CurrentDb

' Open QueryDef object with one parameters.
Set qryDef = m_dbs.QueryDefs("hwy_calc_Preservation_Pavements")
qryDef.Parameters!projID = pid

' Set recordset to new values.
Set m_rst = qryDef.OpenRecordset
' If it is a unique data entry then calculate all scores
If Not m_rst.EOF Then
    Set m_Values = New Collection
    CalculateBenefits
    CalculateOutObjScores
    rtnval = True
End If
SetProjectNumber = rtnval
End Function

Private Sub CalculateBenefits()

'Travel Time Calculations:
'Yearly Travel Time Benefits in Minutes
    AutoTT_Min_I = m_rst("AutoHours_I") * 60
    AutoTT_Min_F = m_rst("AutoHours_F") * 60
    FrTT_Min_I = m_rst("FrHours_I") * 60
    FrTT_Min_F = m_rst("FrHours_F") * 60

'Induced Ridership
    N = m_rst("Fore_Year") - m_rst("Init_Year") + 1
    If IsNumeric(AutoTT_Min_F) And IsNumeric(AutoTT_Min_I) And IsNumeric(N) And N > 0 Then
        LogTT_Min = Log((AutoTT_Min_F / AutoTT_Min_I))
        NPVF_AutoMin = (Exp(LogTT_Min) - 1) / (LogTT_Min / N)
    Else
        NPVF_AutoMin = Null
    End If
    If IsNumeric(FrTT_Min_F) And IsNumeric(FrTT_Min_I) And IsNumeric(N) And N > 0 Then
        LogTT_FrMin = Log((FrTT_Min_F / FrTT_Min_I))
        NPVF_FrMin = (Exp(LogTT_FrMin) - 1) / (LogTT_FrMin / N)
    Else
        NPVF_FrMin = Null
    End If
    AutoTT_Min = AutoTT_Min_I * NPVF_AutoMin
    FrTT_Min = FrTT_Min_I * NPVF_FrMin
    m_TT_Min = AutoTT_Min + FrTT_Min
'Set module level values (to be displayed)
    m_Values.Add m_TT_Min, "TT_MIN"
    m_Values.Add FrTT_Min, "FrTT_MIN"

'Travel Time Benefits in Dollars
    Percent_TV_InVeh = m_rst("Percent_TV_InVeh")
    Time_Value_Veh = m_rst("Time_Value_Veh")
    Time_Value_Freight = m_rst("Time_Value_Truck")
    Discount_Rate = m_rst("Discount_Rate")

```

```

TTAuto_Ben_I = (AutoTT_Min_I) / 60 * (Percent_TV_InVeh * Time_Value_Veh)
TTAuto_Ben_F = (AutoTT_Min_F) / 60 * (Percent_TV_InVeh * Time_Value_Veh)
FrTT_Ben_I = (FrTT_Min_I) / 60 * Time_Value_Freight
FrTT_Ben_F = (FrTT_Min_F) / 60 * Time_Value_Freight
TT_Ben_I = TTAuto_Ben_I + FrTT_Ben_I
TT_Ben_F = TTAuto_Ben_F + FrTT_Ben_F
If IsNumeric(TT_Ben_F) And IsNumeric(TT_Ben_I) And IsNumeric(N) And N > 0 Then
    LogTT_Ben = Log((TT_Ben_F / TT_Ben_I))
    NPVF_TTBen = (Exp(((LogTT_Ben / N) - Discount_Rate) * N) - 1) / _
                ((LogTT_Ben / N) - Discount_Rate)
Else
    NPVF_TTBen = Null
End If
m_TT_Ben = TT_Ben_I * NPVF_TTBen
'Set module level values (to be displayed)
m_Values.Add m_TT_Ben, "TT_BEN"

'Operating Cost Calculations
'User benefit cost calculations
If IsNumeric(User_F) And IsNumeric(User_I) And IsNumeric(N) And N > 0 Then
    LogUserBen = Log((User_F / User_I))
    NPVF_UCBen = (Exp(((LogUserBen / N) - Discount_Rate) * N) - 1) / ((LogUserBen / N) -
Discount_Rate)
Else
    NPVF_UCBen = Null
End If
m_User_Ben = UserBen_I * NPVF_UCBen
'Set module level values (to be displayed)
m_Values.Add m_User_Ben, "USER_BEN"

'Air Pollution
'Emissions Calculations
CO_I = m_rst("CO_I")
CO_F = m_rst("CO_F")
VOC_I = m_rst("VOC_I")
VOC_F = m_rst("VOC_F")
NOX_I = m_rst("NOX_I")
NOX_F = m_rst("NOX_F")
PM10_I = m_rst("PM10_I")
PM10_F = m_rst("PM10_F")
If IsNumeric(CO_F) And IsNumeric(CO_I) And IsNumeric(N) And N > 0 Then
    LogCO = Log((CO_F / CO_I))
    NPVF_CO = (Exp(LogCO) - 1) / (LogCO / N)
Else
    NPVF_CO = Null
End If
If IsNumeric(VOC_F) And IsNumeric(VOC_I) And IsNumeric(N) And N > 0 Then
    LogVOC = Log((VOC_F / VOC_I))
    NPVF_VOC = (Exp(LogVOC) - 1) / (LogVOC / N)
Else
    NPVF_VOC = Null
End If
If IsNumeric(NOX_F) And IsNumeric(NOX_I) And IsNumeric(N) And N > 0 Then
    LogNOX = Log((NOX_F / NOX_I))
    NPVF_NOX = (Exp(LogNOX) - 1) / (LogNOX / N)
Else
    NPVF_NOX = Null
End If
If IsNumeric(PM10_F) And IsNumeric(PM10_I) And IsNumeric(N) And N > 0 Then
    LogPM10 = Log((PM10_F / PM10_I))
    NPVF_PM10 = (Exp(LogPM10) - 1) / (LogPM10 / N)
Else

```

```

    NPVF_PM10 = Null
End If
m_CO_Tons = CO_I * NPVF_CO
m_VOC_Tons = VOC_I * NPVF_VOC
m_NOX_Tons = NOX_I * NPVF_NOX
m_PM10_Tons = NOX_I * NPVF_PM10
'Set module level values (to be displayed)
  m_Values.Add m_CO_Tons, "CO_TONS"
  m_Values.Add m_VOC_Tons, "VOC_TONS"
  m_Values.Add m_NOX_Tons, "NOX_TONS"
  m_Values.Add m_PM10_Tons, "PM10_TONS"

'Emissions Benefit Calculations
If IsNumeric(LogCO) And IsNumeric(N) And N > 0 Then
  NPVF_COBen = (Exp(((LogCO / N) - Discount_Rate) * N) - 1) / ((LogCO / N) - Discount_Rate)
Else
  NPVF_COBen = Null
End If
If IsNumeric(LogVOC) And IsNumeric(N) And N > 0 Then
  NPVF_VOCBen = (Exp(((LogVOC / N) - Discount_Rate) * N) - 1) / ((LogVOC / N) - Discount_Rate)
Else
  NPVF_VOCBen = Null
End If
If IsNumeric(LogNOX) And IsNumeric(N) And N > 0 Then
  NPVF_NOXBen = (Exp(((LogNOX / N) - Discount_Rate) * N) - 1) / ((LogNOX / N) - Discount_Rate)
Else
  NPVF_NOXBen = Null
End If
If IsNumeric(LogPM10) And IsNumeric(N) And N > 0 Then
  NPVF_PM10Ben = (Exp(((LogPM10 / N) - Discount_Rate) * N) - 1) / ((LogPM10 / N) -
Discount_Rate)
Else
  NPVF_PM10Ben = Null
End If
CO_Ben = CO_I * NPVF_COBen
VOC_Ben = VOC_I * NPVF_VOCBen
NOX_Ben = NOX_I * NPVF_NOXBen
PM10_AutoBen = PM10_I * NPVF_PM10Ben
Env_Ben = CO_Ben + VOC_Ben + NOX_Ben + PM10_Ben
m_User_Transfer = 0
'Set module level values (to be displayed)
  m_Values.Add m_Env_Ben, "ENV_BEN"
  m_Values.Add m_User_Transfer, "USER_TRANSFER"

'Safety Calculations
'Accident Calculations (use highway classification as inputted by the user. Variable name "class")
  m_Fatality = 0
  m_Injury = 0
  m_Property = 0
  m_Safety_Ben = 0
'Set module level values (to be displayed)
  m_Values.Add m_Fatality, "FATALITY"
  m_Values.Add m_Injury, "INJURY"
  m_Values.Add m_Property, "PROPERTY"
  m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
tCost = Total_Cost + OpMaint_Cost - Terminal_Cost - EnvRetrofit_Cost
If tCost <> 0 Then
  m_BCR = (m_TT_Ben + m_User_Ben + m_Safety_Ben + m_Env_Ben) / tCost
End If

```

```
m_WSDOT_BCR = (m_TT_Ben + m_User_Ben + m_Safety_Ben + m_Env_Ben) / (WSDOT_Cost + OpMaint - EnvRetrofit)
```

```
'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"
```

```
End Sub
```

```
Private Sub CalculateOutObjScores()
```

```
'Calculation for the System Operation and Maintenance
m_Sys_OM = (m_rst("Q1A") * 34) + (33 * m_rst("Q1B")) + (33 * m_rst("Q1D"))
m_Values.Add CInt(m_Sys_OM), "SYS_OM"
'Calculation for the System Preservation
m_Sys_Pres = 100 * m_rst("Q2A")
m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
m_Sp_Needs = 100 * m_rst("Q3A")
m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
If m_rst("WTP_Corridor") Then
    m_Cong_Rel = m_Values.item("TT_BEN")
Else
    m_Cong_Rel = 0
End If
m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
' This must be updated later
m_Safety = 50
m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
m_Security = 100 * m_rst("Q8A")
m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
m_Commnty = ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
If m_rst("Q10B") > 0 Then
    m_Collab = 50
Else
    m_Collab = 0
End If
m_Collab = m_Collab + (50 * m_rst("Q10A"))
m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
```

```

    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
' This must be updated later
    m_Air_Qual = m_Env_Ben
    m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
                (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = CInt(((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
                    (25 * m_rst("Q16C")) + (25 * m_rst("Q16D")))) / _
                (1 + m_rst("Q16E") + m_rst("Q16F")))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values(itemname)

    GetItemValue = rtnval
End Function

Public Function DisplayFormat(fld As String, dval As Variant) As Variant

    Dim crncylist As String
    Dim intlst As String
    Dim dbllst As String
'String to list the fields with currency formats
    crncylist = "TT_BEN,USER_BEN,USER_TRANSFER,ENV_BEN,SAFETY_BEN," & _
                "TOTAL_COST,WSDOT_COST,FEDERAL_COST"
    intlst = "TT_MIN,SYS_OM,SYS_PRES,SP_NEEDS,CONG_REL,TRAV_OPT,SEAMLESS," & _
            "O_SAFETY,SECURITY,COMMNTY,COLLAB,FREIGHT,ECON_PROS," & _
            "TOURISM,AIR_QUAL,WTR_QUAL,HABITAT,RESOURCE"
    dbllst = "FATALITY,INJURY,PROPERTY,CO_TONS,VOC_TONS,NOX_TONS," & _
            "PM10_TONS,BCR,WSDOT_BCR"
    If InStr(crncylist, fld) <> 0 Then
        If dval <> 0 Then
            dval = Format(dval, "$#,###")
        Else
            dval = "$0"
        End If
    ElseIf InStr(intlst, fld) <> 0 Then
        dval = Round(dval, 0)
    ElseIf InStr(dbllst, fld) <> 0 Then
        dval = Round(dval, 2)
    End If

    DisplayFormat = dval
End Function

Private Function UpdateRstField(ByRef m_rst As DAO.Recordset, fld As String, fldval As Variant) As Boolean
    On Error Resume Next
'Initialize return value
    UpdateRstField = True

```

```

'Edit the fields value
m_rst.Edit
  m_rst(fld) = fldval
m_rst.Update

'If there was an error clear it and return false to indicate that nothing occurred
If Err <> 0 Then
  UpdateRstField = False
  Err.Clear
End If
End Function

Private Sub Class_Terminate()
  Set m_dbsMICA = Nothing
  Set m_rstfryOO = Nothing
  Set m_Values = Nothing
End Sub

```

Structure Preservation Projects

```
Private Sub CalculateBenefits()
```

```

Set m_lkup_Pollution = New lkup_PollutionRate
Set m_lkup_Fatality = New lkup_FatalityRate
Set m_lkup_Shield = New lkup_Shield
Set m_lkup_Speed = New lkup_Speed

```

```
'Load Assumptions
```

```

Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
Discount_Rate = a_rst("Discount_Rate")
Percent_Time_InVeh = a_rst("Percent_TV_InVeh")
Percent_Time_Veh = a_rst("Percent_TV_OutVeh")
Time_Value_Veh = a_rst("Time_Value_Veh")
Time_Value_Freight = a_rst("Time_Value_Truck")
Full_Cost = a_rst("Full_Cost")
Veh_OpCost_Full = a_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = a_rst("Veh_OpCost_Direct")
avo = a_rst("AVO")
CO_Rate_Auto = m_lkup_Pollution.RetrieveRate(AUTOSPEED, em_CO, veht_auto)
If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If
COTon_Cost = a_rst("COTon_Cost")
VOC_Rate_Auto = a_rst("VOC_Rate_Auto")
VOCTon_Cost = a_rst("VOCTon_Cost")
NOX_Rate_Auto = m_lkup_Pollution.RetrieveRate(AUTOSPEED, em_NOX, veht_auto)
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
NOXTon_Cost = a_rst("NOXTon_Cost")
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")
PM10Ton_Cost = a_rst("PM10Ton_Cost")
Fat_Rate_Auto = m_lkup_Fatality.RetrieveRate(2, s_Fatality, veht_auto)
If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
End If
Inj_Rate_Auto = m_lkup_Fatality.RetrieveRate(2, s_Injury, veht_auto)

```

```

If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
End If
Prop_Rate_Auto = m_lkup_Fatality.RetrieveRate(2, s_Property, veht_auto)
If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
End If
Fat_Rate_Truck = m_lkup_Fatality.RetrieveRate(2, s_Fatality, veht_Truck)
If a_rst("Fat_Rate_Truck") <> Fat_Rate_Truck Or IsNull(a_rst("Fat_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Fat_Rate_Truck", Fat_Rate_Truck)
End If
Inj_Rate_Truck = m_lkup_Fatality.RetrieveRate(2, s_Injury, veht_Truck)
If a_rst("Inj_Rate_Truck") <> Inj_Rate_Truck Or IsNull(a_rst("Inj_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Inj_Rate_Truck", Inj_Rate_Truck)
End If
Prop_Rate_Truck = m_lkup_Fatality.RetrieveRate(2, s_Property, veht_Truck)
If a_rst("Prop_Rate_Truck") <> Prop_Rate_Truck Or IsNull(a_rst("Prop_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Prop_Rate_Truck", Prop_Rate_Truck)
End If
Fatality_Cost = a_rst("Fatality_Cost")
Disable_Cost = a_rst("Disable_Cost")
Evident_Cost = a_rst("Evident_Cost")
Possible_Cost = a_rst("Possible_Cost")
PDO_Cost = a_rst("PDO_Cost")

```

```

N = m_rst("Fore_Year") - m_rst("Init_Year") + 1
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ N - 1) / (Discount_Rate * _
  (1 + Discount_Rate) ^ N)) - (N / ((1 + Discount_Rate) ^ N))

```

'Calculate the probability of structure failure

```

RatingWO_I = m_rst("RatingWO_I")
RatingWO_F = m_rst("RatingWO_F")
RatingW_I = m_rst("RatingW_I")
RatingW_F = m_rst("RatingW_F")
If RatingWO_I = 1 Then
  ProbWO_I = 0
Elseif RatingWO_I = 2 Then
  ProbWO_I = 0.25
Elseif RatingWO_I = 3 Then
  ProbWO_I = 0.5
Elseif RatingWO_I = 4 Then
  ProbWO_I = 0.75
Else
  ProbWO_I = 1
End If

```

```

If RatingWO_F = 1 Then
  ProbWO_F = 0
Elseif RatingWO_F = 2 Then
  ProbWO_F = 0.25
Elseif RatingWO_F = 3 Then
  ProbWO_F = 0.5
Elseif RatingWO_F = 4 Then
  ProbWO_F = 0.75
Else

```

```

    ProbWO_F = 1
End If

```

```

If RatingW_I = 1 Then
    ProbW_I = 0
Elseif RatingW_I = 2 Then
    ProbW_I = 0.25
Elseif RatingW_I = 3 Then
    ProbW_I = 0.5
Elseif RatingW_I = 4 Then
    ProbW_I = 0.75
Else
    ProbW_I = 1
End If

```

```

If RatingW_F = 1 Then
    ProbW_F = 0
Elseif RatingW_F = 2 Then
    ProbW_F = 0.25
Elseif RatingW_F = 3 Then
    ProbW_F = 0.5
Elseif RatingW_F = 4 Then
    ProbW_F = 0.75
Else
    ProbW_F = 1
End If

```

```

Prob_I = ProbW_I - ProbWO_I
Prob_F = ProbW_F - ProbWO_F

```

'Travel Time Calculations:

'Yearly Travel Time Benefits in Minutes

```

    TT_Min_I = m_rst("DAMin_I") * avo * Ann_Daily_Benefit * Prob_I
    TT_Min_F = m_rst("DAMin_F") * avo * Ann_Daily_Benefit * Prob_F
    AutoTT_Min_I = TT_Min_I * (1 - m_rst("Freight_I"))
    AutoTT_Min_F = TT_Min_F * (1 - m_rst("Freight_F"))
    FrTT_Min_I = TT_Min_I * m_rst("Freight_I")
    FrTT_Min_F = TT_Min_F * m_rst("Freight_F")
    m_TT_Min = (TT_Min_F - TT_Min_I) * N / 2
    m_FrTT_Min = (FrTT_Min_F - FrTT_Min_I) * N / 2

```

'Set module level values (to be displayed)

```

    m_Values.Add m_TT_Min, "TT_MIN"
    m_Values.Add m_FrTT_Min, "FrTT_MIN"

```

'Travel Time Benefits in Dollars

```

    TTAuto_Ben_I = (AutoTT_Min_I) / 60 * (Percent_TV_InVeh * Time_Value_Veh)
    TTAuto_Ben_F = (AutoTT_Min_F) / 60 * (Percent_TV_InVeh * Time_Value_Veh)
    FrTT_Ben_I = (FrTT_Min_I) / 60 * Time_Value_Freight
    FrTT_Ben_F = (FrTT_Min_F) / 60 * Time_Value_Freight
    TT_Ben_I = TTAuto_Ben_I + FrTT_Ben_I
    TT_Ben_F = TTAuto_Ben_F + FrTT_Ben_F

    TT_BenGrad = (TT_Ben_F - TT_Ben_I) / N

```



```

FrTT_BenGrad = (FrTT_Ben_F - FrTT_Ben_I) / N

m_TT_Ben = TT_Ben_Grad * PofG
m_FrTT_Ben = FrTT_Ben_Grad * PofG
'Set module level values (to be displayed)
  m_Values.Add m_TT_Ben, "TT_BEN"
  m_Values.Add m_FrTT_Ben, "FrTT_BEN"

'Operating Cost Calculations
'User benefit cost calculations
  AutoVMT_I = m_rst("DADist_I") * m_rst("Trips_I") * Ann_Daily_Benefit * Prob_I * (1 -
m_rst("Freight_I"))
  AutoVMT_F = m_rst("DADist_F") * m_rst("Trips_F") * Ann_Daily_Benefit * Prob_F * (1 -
m_rst("Freight_I"))
  FrVMT_I = m_rst("DADist_I") * m_rst("Trips_I") * Ann_Daily_Benefit * Prob_I *
m_rst("Freight_I")
  FrVMT_F = m_rst("DADist_F") * m_rst("Trips_F") * Ann_Daily_Benefit * Prob_F *
m_rst("Freight_I")
  If a_rst("Full_Cost") Then
    User_Ben_I = (AutoVMT_I * Veh_OpCost_Full) + (FrVMT_I * Truck_OpCost_Full)
    User_Ben_F = (AutoVMT_F * Veh_OpCost_Full) + (FrVMT_F * Truck_OpCost_Full)
  Else
    User_Ben_I = (AutoVMT_I * Veh_OpCost_Direct) + (FrVMT_I * Truck_OpCost_Direct)
    User_Ben_F = (AutoVMT_F * Veh_OpCost_Direct) + (FrVMT_F * Truck_OpCost_Direct)
  User_BenGrad = (User_Ben_F - User_Ben_I) / N
  m_User_Ben = User_BenGrad * PofG

'Set module level values (to be displayed)
  m_Values.Add m_User_Ben, "USER_BEN"

'Air Pollution
'Emissions Calculations

'NEED TO ADD IN TRUCK EMISSIONS
CO_Tons_I = (AutoVMT_I * CO_Rate_Auto / 907184.74) + (FrVMT_I * CO_Rate_Truck /
907184.74)
VOC_Tons_I = (AutoVMT_I * VOC_Rate_Auto / 907184.74) + (FrVMT_I * VOC_Rate_Truck
/ 907184.74)
NOX_Tons_I = (AutoVMT_I * NOX_Rate_Auto / 907184.74) + (FrVMT_I * NOX_Rate_Truck
/ 907184.74)
PM10_Tons_I = (AutoVMT_I * PM10_Rate_Auto / 907184.74) + (FrVMT_I *
PM10_Rate_Truck / 907184.74)
CO_Tons_F = (AutoVMT_F * CO_Rate_Auto / 907184.74) + (FrVMT_I * CO_Rate_Truck /
907184.74)
VOC_Tons_F = (AutoVMT_F * VOC_Rate_Auto / 907184.74) + (FrVMT_I *
VOC_Rate_Truck / 907184.74)
NOX_Tons_F = (AutoVMT_F * NOX_Rate_Auto / 907184.74) + (FrVMT_I *
NOX_Rate_Truck / 907184.74)
PM10_Tons_F = (AutoVMT_F * PM10_Rate_Auto / 907184.74) + (FrVMT_I *
PM10_Rate_Truck / 907184.74)
m_CO_Tons = (CO_Tons_F - CO_Tons_I) * N / 2
m_VOC_Tons = (VOC_Tons_F - VOC_Tons_I) * N / 2
m_NOX_Tons = (NOX_Tons_F - NOX_Tons_I) * N / 2
m_PM10_Tons = (PM10_Tons_F - PM10_Tons_I) * N / 2

```

```
'Set module level values (to be displayed)
  m_Values.Add m_CO_Tons, "CO_TONS"
  m_Values.Add m_VOC_Tons, "VOC_TONS"
  m_Values.Add m_NOX_Tons, "NOX_TONS"
  m_Values.Add m_PM10_Tons, "PM10_TONS"
```

'Emissions Benefit Calculations

```
COBen_Grad = (CO_Tons_F - CO_Tons_I) / N * COTon_Cost
CO_Ben = COBen_Grad * PofG
VOCBen_Grad = (VOC_Tons_F - VOC_Tons_I) / N * VOCTon_Cost
VOC_Ben = VOCBen_Grad * PofG
NOXBen_Grad = (NOX_Tons_F - NOX_Tons_I) / N * NOXTon_Cost
NOX_Ben = NOXBen_Grad * PofG
PM10Ben_Grad = (PM10_Tons_F - PM10_Tons_I) / N * PM10Ton_Cost
PM10_Ben = PM10Ben_Grad * PofG
m_Env_Ben = CO_Ben + VOC_Ben + NOX_Ben + PM10_Ben
m_User_Transfer = 0
'Set module level values (to be displayed)
  m_Values.Add m_Env_Ben, "ENV_BEN"
  m_Values.Add m_User_Transfer, "USER_TRANSFER"
```

'Safety Calculations

'Accident Calculations (use highway classification as inputted by the user. Variable name "class")

```
m_Fatality = ((Fat_Rate_Auto * (AutoVMT_F - AutoVMT_I)) + (Fat_Rate_Truck * (FrVMT_F - FrVMT_I))) * N / 2 / 100000000
m_Injury = ((Inj_Rate_Auto * (AutoVMT_F - AutoVMT_I)) + (Inj_Rate_Truck * (FrVMT_F - FrVMT_I))) * N / 2 / 1000000
m_Property = ((PDO_Rate_Auto * (AutoVMT_F - AutoVMT_I)) + (PDO_Rate_Truck * (FrVMT_F - FrVMT_I))) * N / 2 / 1000000
```

```
FatBen_Grad = ((Fat_Rate_Auto * (AutoVMT_F - AutoVMT_I)) + (Fat_Rate_Truck * (FrVMT_F - FrVMT_I))) / N / 100000000 * Fatality_Cost
Fat_Ben = FatBen_Grad * PofG
InjBen_Grad = ((Inj_Rate_Auto * (AutoVMT_F - AutoVMT_I)) + (Inj_Rate_Truck * (FrVMT_F - FrVMT_I))) / N / 1000000 * Evident_Cost
Inj_Ben = InjBen_Grad * PofG
PDOBen_Grad = ((PDO_Rate_Auto * (AutoVMT_F - AutoVMT_I)) + (PDO_Rate_Truck * (FrVMT_F - FrVMT_I))) / N / 1000000 * PDO_Cost
PDO_Ben = PDOBen_Grad * PofG
m_Safety_Ben = Fat_Ben + Inj_Ben + PDO_Ben
'Set module level values (to be displayed)
  m_Values.Add m_Fatality, "FATALITY"
  m_Values.Add m_Injury, "INJURY"
  m_Values.Add m_Property, "PROPERTY"
  m_Values.Add m_Safety_Ben, "SAFETY_BEN"
```

'Cost Calculations

'Capital Cost

```
For i = 1 To 5
  m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * 1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
```

```

    m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * 1 / ((1 + Discount_Rate)
^ (2 * i - 1)))
    m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * 1 / ((1 + Discount_Rate)
^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * 1 / ((1 + Discount_Rate)
^ (2 * i - 1)))
    Next
    m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = WSDOT_annOM * PofA
m_Federal_OM = Federal_annOM * PofA
m_Other1_OM = Other1_annOM * PofA
m_Other2_OM = Other2_annOM * PofA
m_Other3_OM = Other3_annOM * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + m_Other1_Cost + m_Other2_Cost +
m_Other3_Cost

    Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
    m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * 1 / ((1 +
Discount_Rate) ^ (2 * i - 1)))
    m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * 1 / ((1 +
Discount_Rate) ^ (2 * i - 1)))
    m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * 1 / ((1 +
Discount_Rate) ^ (2 * i - 1)))
    Next
    m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations

```

```
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
```

```
m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
```

```
m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
```

```
'Set module level values (to be displayed)
```

```
  m_Values.Add m_Total_Cost, "TOTAL_COST"
```

```
  m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
```

```
  m_Values.Add m_Federal_Cost, "FEDERAL_COST"
```

```
  m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
```

```
  m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
```

```
  m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
```

```
  m_Values.Add m_Other1_Cost, "OTHER1_COST"
```

```
  m_Values.Add m_Other2_Cost, "OTHER2_COST"
```

```
  m_Values.Add m_Other3_Cost, "OTHER3_COST"
```

```
  m_Values.Add m_Cap_Cost, "CAP_COST"
```

```
  m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
```

```
  m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
```

```
  m_Values.Add m_Other1_Cap, "OTHER1_CAP"
```

```
  m_Values.Add m_Other2_Cap, "OTHER2_CAP"
```

```
  m_Values.Add m_Other3_Cap, "OTHER3_CAP"
```

```
  m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
```

```
  m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
```

```
  m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
```

```
  m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
```

```
  m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
```

```
  m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
```

```
  m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
```

```
  m_Values.Add m_WSDOT_OM, "WSDOT_OM"
```

```
  m_Values.Add m_Federal_OM, "FEDERAL_OM"
```

```
  m_Values.Add m_Other1_OM, "OTHER1_OM"
```

```
  m_Values.Add m_Other2_OM, "OTHER2_OM"
```

```
  m_Values.Add m_Other3_OM, "OTHER3_OM"
```

```
  m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
```

```
  m_Values.Add m_BCR, "BCR"
```

```
  m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"
```

```
End Sub
```

```
Private Sub CalculateOutObjScores()
```

```
  'Calculation for the System Operation and Maintenance
```

```
    m_Sys_OM = (m_rst("Q1A") * 34) + (33 * m_rst("Q1B")) + (33 * m_rst("Q1D"))
```

```
    m_Values.Add m_Sys_OM, "SYS_OM"
```

```
  'Calculation for the System Preservation
```

```
    m_Sys_Pres = 100 * m_rst("Q2A")
```

```
    m_Values.Add m_Sys_Pres, "SYS_PRES"
```

```
  'Calculation for the Special Needs Transportation
```

```
    m_Sp_Needs = 100 * m_rst("Q3A")
```

```
    m_Values.Add m_Sp_Needs, "SP_NEEDS"
```

```
  'Calculation for the Congestion Relief
```

```
    If m_rst("WTP_Corridor") Then
```

```
      m_Cong_Rel = m_rst("Q4") * 100
```

```
    Else
```

```
      m_Cong_Rel = 0
```

```
    End If
```

```
    m_Values.Add m_Cong_Rel, "CONG_REL"
```

```

'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  m_Safety = (m_rst("Q7A") * 50) + (m_rst("Q7B") * 50)
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  m_Commnty = -((20 * m_rst("Q9A")) + (20 * m_rst("Q9B"))) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E"))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") > 0 Then
    m_Collab = 50
  Else
    m_Collab = 0
  End If
  m_Collab = m_Collab + (50 * m_rst("Q10A"))
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
  If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + (50 * m_rst("Q14A"))
  Else
    m_Air_Qual = 0
  End If
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
    (33 * m_rst("Q15C"))
  m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
  m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B"))) + _
    (25 * m_rst("Q16C")) + (25 * m_rst("Q16D")) / _
    (1 + m_rst("Q16E") + m_rst("Q16F"))
  m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
  m_Resource = 100 * m_rst("Q17")
  m_Values.Add m_Resource, "RESOURCE"

```

End Sub

Program Code for Highway Safety Projects

(Program code is the same for all project types)

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim a_rst As DAO.Recordset

Dim m_qryName As String

Dim m_calcsComplete As Boolean

Dim m_Values As ValueCollection

'Variables used in intermediate calculations (variants)

Dim Severity_Rating

Dim Forecast_Period

Dim m_WSDOT_Cap

Dim m_Federal_Cap

Dim m_Other1_Cap

Dim m_Other2_Cap

Dim m_Other3_Cap

Dim m_WSDOT_OM

Dim m_Federal_OM

Dim m_Other1_OM

Dim m_Other2_OM

Dim m_Other3_OM

Dim m_FishBarrier_Ben

Dim m_StormWater_Ben

Dim m_NoiseBarrier_Ben

Dim m_FishBarrier_Cap

Dim m_StormWater_Cap

Dim m_NoiseBarrier_Cap

'Variables to hold the calculated values (variants)

Dim m_Total_Benefit

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_EnvRetrofit_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Other1_Cost

Dim m_Other2_Cost

Dim m_Other3_Cost

Dim m_Cap_Cost

Dim m_OpMaint_Cost

Dim m_Terminal_Cost

```
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
Private Sub Class_Initialize()
    m_qryName = "hwy_calc_Safety_Accident_Corridors"1
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
```

¹ The query name varies according to project type. For the three project types recognized in the database, the associated queries are: "hwy_calc_Safety_Accident_Corridors" (for High Accident Corridors), "hwy_calc_Safety_Accident_Locations" (for High Accident Locations), and "hwy_calc_Safety_AtGrade_Intersections" (for At-Grade Intersections). This is the only line that varies in the program code between the three project types.

```

        atype = "prj_Project_Assumptions"
    Else
        atype = "prj_Global_Assumptions"
        pid = asmptnID
    End If

    'Open up a assumption recordset - depend on if in MICA or not
    Set qryDef = m_dbs.QueryDefs(atype)
    qryDef.Parameters!asmptnID = pid

    ' Set recordset to new values.
    Set a_rst = qryDef.OpenRecordset

    'Calculate all of the values for the project type
    CalculateBenefits
    CalculateOutObjScores

    'Check to see if the calcs are done for this project and set input status accordingly
    If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
    m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

    Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
    'Calculate Forecast Period
    Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")

    'Accident Severity Rating
    Severity_Rating = m_rst("severity_rating")

    'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - _
        (Forecast_Period / ((1 + Discount_Rate) ^ Forecast_Period)))
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

```



```

'Travel Time Savings Calculations
  m_TT_Min = 0
  m_TT_Ben = 0

'User Benefit and Environmental Calculations
  m_User_Ben = 0
  m_Env_Ben = 0

  'Set module level values (to be displayed)
  m_Values.Add m_Forecast_Period, "FORECAST_PERIOD"
  m_Values.Add m_TT_Sav, "TT_MIN"
  m_Values.Add m_TT_Ben, "TT_BEN"
  m_Values.Add m_User_Ben, "USER_BEN"
  m_Values.Add m_Env_Ben, "ENV_BEN"

'Safety Calculations
  Fatality_Cost = a_rst("Fatality_Cost")
  Disable_Cost = a_rst("Disable_Cost")
  Evident_Cost = a_rst("Evident_Cost")
  Possible_Cost = a_rst("Possible_Cost")
  PDO_Cost = a_rst("PDO_Cost")

'Accident reduction due to proposed project
  acc_period = m_rst("end_acc_data") - m_rst("begin_acc_data")

  If acc_period <> 0 Then
    ann_fatality = m_rst("prev_fatality") / acc_period
    ann_disable = m_rst("prev_dis_injury") / acc_period
    ann_evident = m_rst("prev_evdnt_injury") / acc_period
    ann_possible = m_rst("prev_poss_injury") / acc_period
    ann_property = m_rst("prev_property") / acc_period
  End If

  m_Fatality = ann_fatality * Forecast_Period
  m_Injury = (ann_disable + ann_evident + ann_possible) * Forecast_Period
  m_Property = ann_property * Forecast_Period

  'Calculate safety benefits
  m_Safety_Ben = ((ann_fatality * Fatality_Cost) + (ann_disable * Disable_Cost) + _
    (ann_evident * Evident_Cost) + (ann_possible * Possible_Cost) + _
    (ann_property * PDO_Cost)) * PofA

'Set module level values (to be displayed)
  m_Values.Add m_Fatality, "FATALITY"
  m_Values.Add m_Injury, "INJURY"
  m_Values.Add m_Property, "PROPERTY"
  m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
'Capital Cost
For i = 1 To 5
  m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))

```

```

m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
  1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
  1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
  1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
  m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

  Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
  m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

```

```

'Benefit-Cost Calculations
  Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
  If (m_Total_Cost - m_Terminal_Cost) = 0 Then
    m_BCR = 0
  Else
    m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
  End If
  If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
    m_WSDOT_BCR = 0
  Else
    m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
  End If
'Set module level values (to be displayed)
  m_Values.Add m_Total_Cost, "TOTAL_COST"
  m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
  m_Values.Add m_Federal_Cost, "FEDERAL_COST"
  m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
  m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
  m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
  m_Values.Add m_Other1_Cost, "OTHER1_COST"
  m_Values.Add m_Other2_Cost, "OTHER2_COST"
  m_Values.Add m_Other3_Cost, "OTHER3_COST"
  m_Values.Add m_Cap_Cost, "CAP_COST"
  m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
  m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
  m_Values.Add m_Other1_Cap, "OTHER1_CAP"
  m_Values.Add m_Other2_Cap, "OTHER2_CAP"
  m_Values.Add m_Other3_Cap, "OTHER3_CAP"
  m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
  m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
  m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
  m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
  m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
  m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
  m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
  m_Values.Add m_WSDOT_OM, "WSDOT_OM"
  m_Values.Add m_Federal_OM, "FEDERAL_OM"
  m_Values.Add m_Other1_OM, "OTHER1_OM"
  m_Values.Add m_Other2_OM, "OTHER2_OM"
  m_Values.Add m_Other3_OM, "OTHER3_OM"
  m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
  m_Values.Add m_BCR, "BCR"
  m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next
'Calculation for the System Operation and Maintenance
  m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
  m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
  m_Sys_Pres = 100 * m_rst("Q2A")

```

```

    m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
    m_Sp_Needs = 100 * m_rst("Q3A")
    m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
    If m_rst("WTP_Corridor") Then
        m_Cong_Rel = 50 + (m_rst("Q4") * 50)
    Else
        m_Cong_Rel = (m_rst("Q4") * 50)
    End If
    m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
    m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
    m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
    m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
    m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
    If m_Safety_Ben > 0 Then
        m_Safety = 50 + (m_rst("Q7B") * 50)
    Else
        m_Safety = m_rst("Q7B") * 50
    End If
    m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
    'The -1 Multiplication Corrects the negative sign introduced for true
    m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E")))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") < 5 Then
        m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
    Else
        m_Collab = (m_rst("Q10A") * 50) + 50
    End If
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    If m_rst("Q14B") = 1 Then
        m_Air_Qual = 50 + m_rst("Q14A") * 50
    Else
        m_Air_Qual = 0
    End If
    m_Values.Add m_Air_Qual, "AIR_QUAL"

```

```
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
    (33 * m_rst("Q15C"))
  m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
  m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
    (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
    (1 + m_rst("Q16E") + m_rst("Q16F"))
  m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
  m_Resource = 100 * m_rst("Q17")
  m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
  Dim rtnval

  rtnval = m_Values.Retrieve(itemname)

  GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
  a_rst.Close
  m_rst.Close
  m_dbs.Close
  Set a_rst = Nothing
  Set m_rst = Nothing
  Set m_dbs = Nothing
  Set m_Values = Nothing
End Sub
```

Program Code for Intelligent Transportation Systems

IDAS Calculations

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim m_Values As Collection

'Variables used in intermediate calculations (variants)

Dim m_VitalScr

Dim m_NonVitalScr

Dim m_Prob(1 To 5)

Dim m_VMT_PC(1 To 5)

'Variables to hold the calculated values (variants)

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_User_Transfer

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Terminal_Cost

Dim m_BCR

Dim m_WSDOT_BCR

'Variables to hold the outcome objective scores (variants)

Dim m_Sys_OM

Dim m_Sys_Pres

Dim m_Sp_Needs

Dim m_Cong_Rel

Dim m_Trav_Opt

Dim m_Seamless

Dim m_Safety

Dim m_Security

Dim m_Commnty

Dim m_Collab

Dim m_Freight

Dim m_Econ_Proc

Dim m_Tourism

Dim m_Air_Qual

Dim m_Wtr_Qual

Dim m_Habitat

Dim m_Resource

'Speed Assumed for lookup table

Private Const AUTOSPEED = 50

Public Function SetProjectNumber(pid As Long) As Boolean

```

Dim rtnval As Boolean
Dim qryDef As DAO.QueryDef

'Initialize return value to false - set true if one record is returned
rtnval = False

Set m_dbs = CurrentDb

' Open QueryDef object with one parameters.
Set qryDef = m_dbs.QueryDefs("its_calc_IDAS")
qryDef.Parameters!projID = pid

' Set recordset to new values.
Set m_rst = qryDef.OpenRecordset
' If it is a unique data entry then calculate all scores
If Not m_rst.EOF Then
    Set m_Values = New Collection
    CalculateBenefits
    CalculateOutObjScores
    rtnval = True
End If
SetProjectNumber = rtnval
End Function

Private Sub CalculateBenefits()

'Travel Time Calculations:
'Yearly Travel Time Benefits in Minutes
    Freight_I = m_rst("Freight_I")
    Freight_F = m_rst("Freight_F")
    Hours_BCI = m_rst("Hours_BCI")
    Hours_BCF = m_rst("Hours_BCF")
    Hours_PCI = m_rst("Hours_PCI")
    Hours_PCF = m_rst("Hours_PCF")
    Trips_BCI = m_rst("Trips_BCI")
    Trips_BCF = m_rst("Trips_BCF")
    Trips_PCI = m_rst("Trips_PCI")
    Trips_PCF = m_rst("Trips_PCF")
    TT_Min_I = (Hours_PCI - Hours_BCI) * 60
    TT_Min_F = (Hours_PCF - Hours_PCF) * 60
    FrTT_Min_I = (Hours_PCI - Hours_BCI) * 60 * Freight_I
    FrTT_Min_F = (Hours_PCF - Hours_BCF) * 60 * Freight_F
    AutoTrips_I = (Trips_PCI - Trips_BCI) * (1 - Freight_I)
    AutoTrips_F = (Trips_PCF - Trips_BCF) * (1 - Freight_F)
    FrTrips_I = (Trips_PCI - Trips_BCI) * Freight_I
    FrTrips_F = (Trips_PCF - Trips_BCF) * Freight_F

'Induced Ridership
    N = m_rst("Fore_Year") - m_rst("Init_Year") + 1
    If IsNumeric(TT_Min_F) And IsNumeric(TT_Min_I) And IsNumeric(N) And N > 0 Then
        LogTT_Min = Log((TT_Min_F / TT_Min_I))
        NPVF_Min = (Exp(((LogTT_Min / N)) * N) - 1) / _
            ((LogTT_Min / N))
    Else
        NPVF_Min = Null
    End If
    If IsNumeric(FrTT_Min_F) And IsNumeric(FrTT_Min_I) And IsNumeric(N) And N > 0 Then
        LogTT_FrMin = Log((FrTT_Min_F / FrTT_Min_I))
        NPVF_FrMin = (Exp(((LogTT_FrMin / N)) * N) - 1) / _

```

```

        ((LogTT_FrMin / N))
    Else
        NPVF_FrMin = Null
    End If
    m_TT_Min = TT_Min_I * NPVF_Min
    FrTT_Min = FrTT_Min_I * NPVF_FrMin
    'Set module level values (to be displayed)
    m_Values.Add m_TT_Min, "TT_MIN"
    m_Values.Add FrTT_Min, "FrTT_MIN"

'Travel Time Benefits in Dollars
Percent_TV_InVeh = m_rst("Percent_TV_InVeh")
Time_Value_Veh = m_rst("Time_Value_Veh")
Time_Value_Freight = m_rst("Time_Value_Truck")
Discount_Rate = m_rst("Discount_Rate")

TTAuto_Ben_I = (TT_Min_I) / 60 * (Percent_TV_InVeh * Time_Value_Veh) * (1 - Freight_I)
TTAuto_Ben_F = (TT_Min_F) / 60 * (Percent_TV_InVeh * Time_Value_Veh) * (1 - Freight_F)
FrTT_Ben_I = (TT_Min_I) / 60 * Time_Value_Freight * Freight_I
FrTT_Ben_F = (TT_Min_F) / 60 * Time_Value_Freight * Freight_F
TT_Ben_I = TTAuto_Ben_I + FrTT_Ben_I
TT_Ben_F = TTAuto_Ben_F + FrTT_Ben_F
If IsNumeric(TT_Ben_F) And IsNumeric(TT_Ben_I) And IsNumeric(N) And N > 0 Then
    LogTT_Ben = Log((TT_Ben_F / TT_Ben_I))
    NPVF_TTBen = (Exp(((LogTT_Ben / N) - Discount_Rate) * N) - 1) / _
                ((LogTT_Ben / N) - Discount_Rate)
Else
    NPVF_TTBen = Null
End If
If IsNumeric(FrTT_Ben_F) And IsNumeric(FrTT_Ben_I) And IsNumeric(N) And N > 0 Then
    LogFrTT_Ben = Log((FrTT_Ben_F / FrTT_Ben_I))
    NPVF_FrTTBen = (Exp(((LogFrTT_Ben / N) - Discount_Rate) * N) - 1) / _
                ((LogFrTT_Ben / N) - Discount_Rate)
Else
    NPVF_TTBen = Null
End If
m_TT_Ben = TT_Ben_I * NPVF_TTBen
FrTT_Ben = FrTT_Ben_I * NPVF_FrTTBen
'Set module level values (to be displayed)
m_Values.Add m_TT_Ben, "TT_BEN"
m_Values.Add FrTT_Ben, "FrTT_BEN"

'Operating Cost Calculations
VMT_BCI = m_rst("VMT_BCI")
VMT_PCI = m_rst("VMT_PCI")
VMT_BCF = m_rst("VMT_BCF")
VMT_PCF = m_rst("VMT_PCF")
VMTAuto_I = (VMT_PCI - VMT_BCI) * (1 - Freight_I)
VMTAuto_F = (VMT_PCF - VMT_BCF) * (1 - Freight_F)
FrVMT_I = (VMT_PCI - VMT_BCI) * Freight_I
FrVMT_F = (VMT_PCF - VMT_BCF) * Freight_F
If IsNumeric(VMTAuto_I) And IsNumeric(VMTAuto_F) And IsNumeric(N) And N > 0 Then
    LogVMTAuto = Log((VMTAuto_F / VMTAuto_I))
    NPVF_VMTAuto = ((Exp(LogVMTAuto) - 1) / (LogVMTAuto) / N)
Else
    NPVF_VMTAuto = Null
End If
If IsNumeric(FrVMT_F) And IsNumeric(FrVMT_I) And IsNumeric(N) And N > 0 Then
    LogFrVMT = Log((FrVMT_F / FrVMT_I))
    NPVF_FrVMT = ((Exp(LogFrVMT) - 1) / (LogFrVMT) / N)
Else
    NPVF_FrVMT = Null

```



```

End If
VMTAuto_Tot = VMTAuto_I * NPVFAuto_VMT
FrVMT_Tot = FrVMT_I * NPVF_FrVMT

'User benefit cost calculations
Veh_OpCost_Full = m_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = m_rst("Veh_OpCost_Direct")
Freight_OpCost = m_rst("Freight_OpCost")
If m_rst("Full_Cost") Then
    UserBen_I = (VMTAuto_I * Veh_OpCost_Full) + (FrVMT_I * Freight_OpCost)
    UserBen_F = (VMTAuto_F * Veh_OpCost_Full) + (FrVMT_F * Freight_OpCost)
Else
    UserBen_I = (VMTAuto_I * Veh_OpCost_Direct) + (FrVMT_I * Freight_OpCost)
    UserBen_F = (VMTAuto_F * Veh_OpCost_Direct) + (FrVMT_F * Freight_OpCost)
End If
If IsNumeric(UserBen_F) And IsNumeric(UserBen_I) And IsNumeric(N) And N > 0 Then
    LogUserBen = Log((UserBen_F / UserBen_I))
    NPVF_UCBen = (Exp(((LogUserBen / N) - Discount_Rate) * N) - 1) / ((LogUserBen / N) -
Discount_Rate)
Else
    NPVF_UCBen = Null
End If
m_User_Ben = UserBen_I * NPVF_UCBen
'Set module level values (to be displayed)
m_Values.Add m_User_Ben, "USER_BEN"
'Air Pollution
'Emissions Calculations
Per_Cold_Auto = m_rst("Per_Cold_Auto")
Per_Cold_Truck = m_rst("Per_Cold_Truck")
CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)
If m_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(m_rst("CO_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If
CO_Rate_Truck = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_truck)
If m_rst("CO_Rate_Truck") <> NOX_Rate_Truck Or IsNull(m_rst("CO_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "CO_Rate_Truck", CO_Rate_Truck)
End If
COTon_Cost = m_rst("COTon_Cost")
VOC_Rate_Auto = m_rst("VOC_Rate_Auto")
VOC_Rate_Truck = m_rst("VOC_Rate_Truck")
VOCTon_Cost = m_rst("VOCTon_Cost")
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
If m_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(m_rst("NOX_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
NOX_Rate_Truck = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_truck)
If m_rst("NOX_Rate_Truck") <> NOX_Rate_Truck Or IsNull(m_rst("NOX_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "NOX_Rate_Truck", NOX_Rate_Truck)
End If
NOXTon_Cost = m_rst("NOXTon_Cost")
PM10_Rate_Auto = m_rst("PM10_Rate_Auto")
PM10_Rate_Truck = m_rst("PM10_Rate_Truck")
PM10Ton_Cost = m_rst("PM10Ton_Cost")

CO_AutoTons = (VMTAuto_I * NPVF_VMTAuto * CO_Rate_Auto * (1 / 1000) * (0.9842 / 1000)) + _
(AutoTrips_I * NPVF_VMTAuto * Per_Cold_Auto * m_rst("CO_Cold_Auto") * (1 / 1000) *
(0.9842 / 1000))
VOC_AutoTons = (VMTAuto_I * NPVF_VMTAuto * VOC_Rate_Auto * (1 / 1000) * (0.9842 / 1000))
NOX_AutoTons = (VMTAuto_I * NPVF_VMTAuto * NOX_Rate_Auto * (1 / 1000) * (0.9842 / 1000)) + _
(AutoTrips_I * NPVF_VMTAuto * Per_Cold_Auto * m_rst("NOX_Cold_Auto") * (1 / 1000) *
(0.9842 / 1000))
PM10_AutoTons = VMTAuto_I * NPVF_VMTAuto * PM10_Rate_Auto * (1 / 1000) * (0.9842 / 1000) + _

```

```

      (AutoTrips_I * NPVF_VMTAuto * Per_Cold_Auto * m_rst("PM10_Cold_Auto") * (1 / 1000) *
(0.9842 / 1000))
      CO_FrTons = FrVMT_I * NPVF_FrVMT * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) +
      (FrTrips_I * NPVF_FrVMT * Per_Cold_Truck * m_rst("CO_Cold_Truck") * (1 / 1000) * (0.9842 /
1000))
      VOC_FrTons = FrVMT_I * NPVF_FrVMT * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000)
      NOX_FrTons = FrVMT_I * NPVF_FrVMT * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) +
      (FrTrips_I * NPVF_FrVMT * Per_Cold_Truck * m_rst("NOX_Cold_Truck") * (1 / 1000) * (0.9842
/ 1000))
      PM10_FrTons = FrVMT_I * NPVF_FrVMT * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000) +
      (FrTrips_I * NPVF_FrVMT * Per_Cold_Truck * m_rst("PM10_Cold_Truck") * (1 / 1000) * (0.9842
/ 1000))
      m_CO_Tons = CO_AutoTons + CO_FrTons
      m_VOC_Tons = VOC_AutoTons + VOC_FrTons
      m_NOX_Tons = NOX_AutoTons + NOX_FrTons
      m_PM10_Tons = PM10_AutoTons + PM10_FrTons
      'Set module level values (to be displayed)
      m_Values.Add m_CO_Tons, "CO_TONS"
      m_Values.Add m_VOC_Tons, "VOC_TONS"
      m_Values.Add m_NOX_Tons, "NOX_TONS"
      m_Values.Add m_PM10_Tons, "PM10_TONS"

      'Emissions Benefit Calculations
      If IsNumeric(VMTAuto_F) And IsNumeric(VMTAuto_I) And IsNumeric(N) And N > 0 Then
        LogVMTAuto = Log((VMTAuto_F / VMTAuto_I))
        NPVF_AutoEnvBen = (Exp(((LogVMTAuto / N) - Discount_Rate) * N) - 1) / ((LogVMTAuto / N) -
Discount_Rate)
      Else
        NPVF_AutoEnvBen = Null
      End If
      If IsNumeric(FrVMT_F) And IsNumeric(FrVMT_I) And IsNumeric(N) And N > 0 Then
        LogFrVMT = Log((FrVMT_F / FrVMT_I))
        NPVF_FrEnvBen = (Exp(((LogFrVMT / N) - Discount_Rate) * N) - 1) / ((LogFrVMT / N) -
Discount_Rate)
      Else
        NPVF_FrEnvBen = Null
      End If

      CO_AutoBen = VMTAuto_I * CO_Rate_Auto * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_AutoEnvBen
      VOC_AutoBen = VMTAuto_I * VOC_Rate_Auto * (1 / 1000) * (0.9842 / 1000) * VOCTon_Cost *
NPVF_AutoEnvBen
      NOX_AutoBen = VMTAuto_I * NOX_Rate_Auto * (1 / 1000) * (0.9842 / 1000) * NOXTon_Cost *
NPVF_AutoEnvBen
      PM10_AutoBen = VMTAuto_I * PM10_Rate_Auto * (1 / 1000) * (0.9842 / 1000) * PM10Ton_Cost *
NPVF_AutoEnvBen
      CO_FrBen = FrVMT_I * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_FrEnvBen
      VOC_FrBen = FrVMT_I * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * VOCTon_Cost *
NPVF_FrEnvBen
      NOX_FrBen = FrVMT_I * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * NOXTon_Cost *
NPVF_FrEnvBen
      PM10_FrBen = FrVMT_I * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * PM10Ton_Cost *
NPVF_FrEnvBen

      m_Env_Ben = CO_AutoBen + VOC_AutoBen + NOX_AutoBen + PM10_AutoBen + CO_FrBen +
VOC_FrBen + NOX_FrBen + PM10_FrBen

      m_User_Transfer = 0
      'Set module level values (to be displayed)
      m_Values.Add m_Env_Ben, "ENV_BEN"
      m_Values.Add m_User_Transfer, "USER_TRANSFER"

```

```

'Safety Calculations
'Accident Calculations (use highway classification as inputted by the user. Variable name "class")
  Dim hclass As Integer
  If IsNumeric(m_rst("Class")) Then
    hclass = m_rst("Class")
  Else
    hclass = 2
  End If
  Fat_Rate_Auto = LookupFatalityRate(m_dbs, hclass, s_Fatality, veht_auto)
  If m_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(m_rst("Fat_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
  End If
  Inj_Rate_Auto = LookupFatalityRate(m_dbs, hclass, s_Injury, veht_auto)
  If m_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(m_rst("Inj_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
  End If
  Prop_Rate_Auto = LookupFatalityRate(m_dbs, hclass, s_Property, veht_auto)
  If m_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(m_rst("Prop_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
  End If
  Fat_Rate_Truck = LookupFatalityRate(m_dbs, hclass, s_Fatality, veht_truck)
  If m_rst("Fat_Rate_Auto") <> Fat_Rate_Truck Or IsNull(m_rst("Fat_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "Fat_Rate_Truck", Fat_Rate_Truck)
  End If
  Inj_Rate_Truck = LookupFatalityRate(m_dbs, hclass, s_Injury, veht_truck)
  If m_rst("Inj_Rate_Truck") <> Inj_Rate_Truck Or IsNull(m_rst("Inj_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "Inj_Rate_Truck", Inj_Rate_Truck)
  End If
  Prop_Rate_Truck = LookupFatalityRate(m_dbs, hclass, s_Property, veht_truck)
  If m_rst("Prop_Rate_Truck") <> Prop_Rate_Truck Or IsNull(m_rst("Prop_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "Prop_Rate_Truck", Prop_Rate_Truck)
  End If
  Fatality_Auto = VMTAuto_I * NPVF_VMTAuto * Fat_Rate_Auto / 100000000
  Injury_Auto = VMTAuto_I * NPVF_VMTAuto * Inj_Rate_Auto / 1000000
  Property_Auto = VMTAuto_I * NPVF_VMTAuto * Prop_Rate_Auto / 1000000
  Fatality_Fr = FrVMT_I * NPVF_FrVMT * Fat_Rate_Truck / 100000000
  Injury_Fr = FrVMT_I * NPVF_FrVMT * Inj_Rate_Truck / 1000000
  Property_Fr = FrVMT_I * NPVF_FrVMT * Prop_Rate_Truck / 1000000
  m_Fatality = Fatality_Auto + Fatality_Fr
  m_Injury = Injury_Auto + Injury_Fr
  m_Property = Property_Auto + Property_Fr
'Set module level values (to be displayed)
  m_Values.Add m_Fatality, "FATALITY"
  m_Values.Add m_Injury, "INJURY"
  m_Values.Add m_Property, "PROPERTY"

'Safety Benefit Calculations (use highway classification as inputted by the user. Variable name "class")
  NPVF_AutoSafety = NPVF_AutoEnvBen
  Fatality_AutoBen = VMTAuto_I * Fat_Rate_Auto / 100000000 * Cost_Fatality * NPVF_AutoSafety
  Injury_AutoBen = VMTAuto_I * Inj_Rate_Auto / 1000000 * Cost_Evident * NPVF_AutoSafety
  Prop_AutoBen = VMTAuto_I * Prop_Rate_Auto / 1000000 * Cost_PDO * NPVF_AutoSafety
  NPVF_FrSafety = NPVF_FrEnvBen
  Fatality_FrBen = VMTAuto_I * Fat_Rate_Truck / 100000000 * Cost_Fatality * NPVF_FrSafety
  Injury_FrBen = VMTAuto_I * Inj_Rate_Truck / 1000000 * Cost_Evident * NPVF_FrSafety
  Prop_FrBen = VMTAuto_I * Prop_Rate_Truck / 1000000 * Cost_PDO * NPVF_FrSafety

  m_Safety_Ben = Fatality_AutoBen + Injury_AutoBen + Prop_AutoBen + Fatality_FrBen + _
    Injury_FrBen + Prop_FrBen
'Set module level values (to be displayed)
  m_Values.Add m_Safety_Ben, "SAFETY_BEN"

```

```

'Cost Calculations
tCost = Total_Cost + OpMaint_Cost - Terminal_Cost - EnvRetrofit_Cost
If tCost <> 0 Then
    m_BCR = (m_TT_Ben + m_User_Ben + m_Safety_Ben + m_Env_Ben) / tCost
End If
m_WSDOT_BCR = (m_TT_Ben + m_User_Ben + m_Safety_Ben + m_Env_Ben) / (WSDOT_Cost +
OpMaint - EnvRetrofit)

'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

End Sub

Private Sub CalculateOutObjScores()

'Calculation for the System Operation and Maintenance
m_Sys_OM = (m_rst("Q1A") * 34) + (33 * m_rst("Q1B")) + (33 * m_rst("Q1D"))
m_Values.Add CInt(m_Sys_OM), "SYS_OM"
'Calculation for the System Preservation
m_Sys_Pres = 100 * m_rst("Q2A")
m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
m_Sp_Needs = 100 * m_rst("Q3A")
m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
If m_rst("WTP_Corridor") Then
    m_Cong_Rel = m_Values.item("TT_BEN")
Else
    m_Cong_Rel = 0
End If
m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
' This must be updated later
m_Safety = m_Safety_Ben
m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
m_Security = 100 * m_rst("Q8A")
m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
m_Commnty = ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
(20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
(20 * m_rst("Q9E")))
m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
If m_rst("Q10B") > 0 Then
    m_Collab = 50
Else
    m_Collab = 0
End If
m_Collab = m_Collab + (50 * m_rst("Q10A"))
m_Values.Add m_Collab, "COLLAB"

```

```

'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
  ' This must be updated later
  m_Air_Qual = m_Env_Ben
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
    (33 * m_rst("Q15C"))
  m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
  m_Habitat = CInt(((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
    (25 * m_rst("Q16C")) + (25 * m_rst("Q16D")))) / _
    (1 + m_rst("Q16E") + m_rst("Q16F")))
  m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
  m_Resource = 100 * m_rst("Q17")
  m_Values.Add m_Resource, "RESOURCE"

```

End Sub

Public Function GetItemValue(itemname As String) As Variant

Dim rtnval

rtnval = m_Values(itemname)

GetItemValue = rtnval

End Function

Public Function DisplayFormat(fld As String, dval As Variant) As Variant

Dim crncylist As String

Dim intlist As String

Dim dbllist As String

'String to list the fields with currency formats

crncylist = "TT_BEN,USER_BEN,USER_TRANSFER,ENV_BEN,SAFETY_BEN," & _
"TOTAL_COST,WSDOT_COST,FEDERAL_COST"

intlist = "TT_MIN,SYS_OM,SYS_PRES,SP_NEEDS,CONG_REL,TRAV_OPT,SEAMLESS," & _
"O_SAFETY,SECURITY,COMMNTY,COLLAB,FREIGHT,ECON_PROS," & _
"TOURISM,AIR_QUAL,WTR_QUAL,HABITAT,RESOURCE"

dbllist = "FATALITY,INJURY,PROPERTY,CO_TONS,VOC_TONS,NOX_TONS," & _
"PM10_TONS,BCR,WSDOT_BCR"

If InStr(crncylist, fld) <> 0 Then

 If dval <> 0 Then

 dval = Format(dval, "\$#,###")

 Else

 dval = "\$0"

 End If

Elseif InStr(intlist, fld) <> 0 Then

 dval = Round(dval, 0)

Elseif InStr(dbllist, fld) <> 0 Then

 dval = Round(dval, 2)

End If

DisplayFormat = dval

End Function

```
Private Function UpdateRstField(ByRef m_rst As DAO.Recordset, fld As String, fldval As Variant) As Boolean
```

```
    On Error Resume Next
    'Initialize return value
    UpdateRstField = True
```

```
    'Edit the fields value
    m_rst.Edit
    m_rst(fld) = fldval
    m_rst.Update
```

```
    'If there was an error clear it and return false to indicate that nothing occurred
    If Err <> 0 Then
        UpdateRstField = False
        Err.Clear
    End If
```

End Function

```
Private Sub Class_Terminate()
```

```
    Set m_dbsMICA = Nothing
    Set m_rstfryOO = Nothing
    Set m_Values = Nothing
```

End Sub

SCRITS Calculations

```
VERSION 1.0 CLASS
```

```
BEGIN
```

```
    MultiUse = -1 'True
```

```
END
```

```
Attribute VB_Name = "its_calcSCRITS"
```

```
Attribute VB_GlobalNameSpace = False
```

```
Attribute VB_Creatable = False
```

```
Attribute VB_PredeclaredId = False
```

```
Attribute VB_Exposed = False
```

```
Option Compare Database
```

```
'Variables for the class level database objects
```

```
Dim m_dbs As DAO.Database
```

```
Dim m_rst As DAO.Recordset
```

```
Dim a_rst As DAO.Recordset
```

```
Dim m_qryName As String
```

```
Dim m_calcsComplete As Boolean
```

```
Dim m_Values As ValueCollection
```

```
Dim m_P_ID As Integer
```

```
'Variables to hold lookup functions
```

```
Dim m_ikup_Pollution As Ikup_PollutionRate
```

```
Dim m_ikup_Fatality As Ikup_FatalityRate
```

```
Dim m_ikup_Shield As Ikup_Shield
```

```
Dim m_ikup_Speed As Ikup_Speed
```

```
'Speed Assumed for lookup table
```

```
Private Const AUTOSPEED = 50
```

```
Private Sub Class_Initialize()
```

```
    m_qryName = "its_calc_SCRITS"
```

```
    m_calcsComplete = False
```

End Sub

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```

    Dim atype As String
    Dim qryDef As DAO.QueryDef

    Set m_dbs = CurrentDb

    m_P_ID = pid

    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid

    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then

        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection

        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If

        'Open up a assumption recordset - depend on if in MICA or not
        Set qryDef = m_dbs.QueryDefs(atype)
        qryDef.Parameters!asmptnID = pid

        ' Set recordset to new values.
        Set a_rst = qryDef.OpenRecordset

        'Calculate all of the values for the project type
        CalculateBenefits
        Call SetSCRITSType("its_calc_SCRITS")
        CalculateOutObjScores

        'Check to see if the calcs are done for this project and set input status accordingly
        If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
        m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

        Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

    End If

    'Close the Querydef object
    qryDef.Close
    Set qryDef = Nothing

    CalculateProjectType = m_calcsComplete
End Function

Private Sub SetSCRITSType(m_qryName As String)

    If Not IsNumeric(m_P_ID) Then Exit Sub
    Dim qryDef As DAO.QueryDef

```

```

Set m_dbs = CurrentDb

' Open QueryDef object with one parameters.
Set qryDef = m_dbs.QueryDefs(m_qryName)
qryDef.Parameters!projID = m_P_ID

' Set recordset to new values.
Set m_rst = qryDef.OpenRecordset

qryDef.Close
Set qryDef = Nothing

End Sub

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()

Set m_ikup_Pollution = New Ikup_PollutionRate
Set m_ikup_Fatality = New Ikup_FatalityRate
Set m_ikup_Shield = New Ikup_Shield
Set m_ikup_Speed = New Ikup_Speed

'Load Assumptions
Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
Discount_Rate = a_rst("Discount_Rate")
Percent_Time_InVeh = a_rst("Percent_TV_InVeh")
Percent_Time_Veh = a_rst("Percent_TV_OutVeh")
Time_Value_Veh = a_rst("Time_Value_Veh")
Time_Value_Freight = a_rst("Time_Value_Truck")
Full_Cost = a_rst("Full_Cost")
Veh_OpCost_Full = a_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = a_rst("Veh_OpCost_Direct")
avo = a_rst("AVO")
CO_Rate_Auto = m_ikup_Pollution.RetrieveRate(AUTOSPEED, em_CO, veht_auto)
If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If
COTon_Cost = a_rst("COTon_Cost")
VOC_Rate_Auto = a_rst("VOC_Rate_Auto")
VOCTon_Cost = a_rst("VOCTon_Cost")
NOX_Rate_Auto = m_ikup_Pollution.RetrieveRate(AUTOSPEED, em_NOX, veht_auto)
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
NOXTon_Cost = a_rst("NOXTon_Cost")
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")
PM10Ton_Cost = a_rst("PM10Ton_Cost")
Fat_Rate_Auto = m_ikup_Fatality.RetrieveRate(2, s_Fatality, veht_auto)
If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
End If
Inj_Rate_Auto = m_ikup_Fatality.RetrieveRate(2, s_Injury, veht_auto)
If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)

```



```

End If
Prop_Rate_Auto = m_Ikup_Fatality.RetrieveRate(2, s_Property, veht_auto)
If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
End If
Fat_Rate_Truck = m_Ikup_Fatality.RetrieveRate(2, s_Fatality, veht_Truck)
If a_rst("Fat_Rate_Truck") <> Fat_Rate_Truck Or IsNull(a_rst("Fat_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Fat_Rate_Truck", Fat_Rate_Truck)
End If
Inj_Rate_Truck = m_Ikup_Fatality.RetrieveRate(2, s_Injury, veht_Truck)
If a_rst("Inj_Rate_Truck") <> Inj_Rate_Truck Or IsNull(a_rst("Inj_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Inj_Rate_Truck", Inj_Rate_Truck)
End If
Prop_Rate_Truck = m_Ikup_Fatality.RetrieveRate(2, s_Property, veht_Truck)
If a_rst("Prop_Rate_Truck") <> Prop_Rate_Truck Or IsNull(a_rst("Prop_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Prop_Rate_Truck", Prop_Rate_Truck)
End If
Fatality_Cost = a_rst("Fatality_Cost")
Disable_Cost = a_rst("Disable_Cost")
Evident_Cost = a_rst("Evident_Cost")
Possible_Cost = a_rst("Possible_Cost")
PDO_Cost = a_rst("PDO_Cost")

'Baseline Calculations:
If m_rst("RecVHT_Input") = 1 Then
  RecFreeVHT_BCI = m_rst("FreeACR_BCI") / m_Ikup_Speed.RetrieveRate(m_rst("FreeACR_BCI"),
stFree)
  RecFreeVHT_BCF = m_rst("FreeACR_BCF") / m_Ikup_Speed.RetrieveRate(m_rst("FreeACR_BCF"),
stFree)
  RecArtVHT_BCI = m_rst("ArtACR_BCI") / m_Ikup_Speed.RetrieveRate(m_rst("ArtACR_BCI"), stFree)
  RecArtVHT_BCF = m_rst("ArtACR_BCF") / m_Ikup_Speed.RetrieveRate(m_rst("ArtACR_BCF"), stFree)
Else
  RecFreeVHT_BCI = m_rst("RecFreeVHT_BCI")
  RecFreeVHT_BCF = m_rst("RecFreeVHT_BCF")
  RecArtVHT_BCI = m_rst("RecArtVHT_BCI")
  RecArtVHT_BCF = m_rst("RecArtVHT_BCF")
End If
' Need to define LookupIncPer_Shld and LookupIncPer_NoShld. Tables already imported.
IncPer_Shld_BCI = m_Ikup_Shield.RetrieveRate(FreeACR_BCI, stYes)
IncPer_Shld_BCF = m_Ikup_Shield.RetrieveRate(FreeACR_BCF, stYes)
IncPer_NoShld_BCI = m_Ikup_Shield.RetrieveRate(FreeACR_BCI, stNo)
IncPer_NoShld_BCF = m_Ikup_Shield.RetrieveRate(FreeACR_BCF, stNo)
If m_rst("NonRecVHT_Input") = 1 Then
  RatioVHT_BCI = (IncPer_Shld_BCI * m_rst("Free_ShldPer")) + (IncPer_NoShld_BCI * (1 -
m_rst("Free_ShldPer")))
  RatioVHT_BCF = (IncPer_Shld_BCF * m_rst("Free_ShldPer")) + (IncPer_NoShld_BCF * (1 -
m_rst("Free_ShldPer")))
Else
  RatioVHT_BCI = m_rst("RatioVHT_BCI")
  RatioVHT_BCF = m_rst("RatioVHT_BCF")
End If
AvgFree_Rec_BCI = m_rst("FreeVMT_BCI") / RecFreeVHT_BCI
AvgFree_Rec_BCF = m_rst("FreeVMT_BCF") / RecFreeVHT_BCF
AvgArt_Rec_BCI = m_rst("ArtVMT_BCI") / RecArtVHT_BCI
AvgArt_Rec_BCF = m_rst("ArtVMT_BCF") / RecArtVHT_BCF
NonRecVHT_BCI = RatioVHT_BCI * RecFreeVHT_BCI
NonRecVHT_BCF = RatioVHT_BCF * RecFreeVHT_BCF
AvgFree_Tot_BCI = m_rst("FreeVMT_BCI") / (RecFreeVHT_BCI + NonRecVHT_BCI)
AvgFree_Tot_BCF = m_rst("FreeVMT_BCF") / (RecFreeVHT_BCF + NonRecVHT_BCF)
AvgArt_Tot_BCI = AvgArt_Rec_BCI
AvgArt_Tot_BCF = AvgArt_Rec_BCF

```

```

Avg_IncDur = m_rst("Avg_IncDur")
FreeVMT_BCI = m_rst("FreeVMT_BCI")
FreeVMT_BCF = m_rst("FreeVMT_BCF")
SecAccPer = m_rst("SecAccPer")
AvgTripNo_BCI = m_rst("Avg_TripNo_BCI")
AvgTripNo_BCF = m_rst("Avg_TripNo_BCF")
Avg_TripMin_BCI = m_rst("Avg_TripMin_BCI")
Avg_TripMin_BCF = m_rst("Avg_TripMin_BCF")
Avg_TripLen_BCI = m_rst("Avg_TripLen_BCI")
Avg_TripLen_BCF = m_rst("Avg_TripLen_BCF")
Avg_BusFare = m_rst("Avg_BusFare")
ArtVMT_BCI = m_rst("ArtVMT_BCI")
ArtVMT_BCF = m_rst("ArtVMT_BCF")
ArtSpeed_BCI = m_lkup_Speed.RetrieveRate(m_rst("ArtACR_BCI"), stArterial)
ArtSpeed_BCF = m_lkup_Speed.RetrieveRate(m_rst("ArtACR_BCF"), stArterial)
N = m_rst("Fore_Year") - m_rst("Init_Year") + 1
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ N - 1) / (Discount_Rate * (1 + Discount_Rate) ^ N)) -
(N / ((1 + Discount_Rate) ^ N)))

```

```

bCCTV = m_rst("CCTV")
bDET = m_rst("FreeDet")
bHAR = m_rst("HAR")
bVMS = m_rst("VMS")
bATIS = m_rst("ATIS")
bKio = m_rst("Kiosks")
bCVOKio = m_rst("CVOKiosks")
bInt = m_rst("Internet")
bAVL = m_rst("BusAVL")
bEFC = m_rst("BusEFC")
bBPS = m_rst("BusPS")
bETC = m_rst("ETC")
bRamp = m_rst("RampMetering")
bWIM = m_rst("WeighInMotion")
bRR = m_rst("RRCrossing")
bTSS = m_rst("TrafficSS")

```

If bCCTV Then

 Call SetSCRITStype("its_calc_SCRITS_CCTV")

 If Not m_rst.EOF Then

 '-----
 'Closed Circuit TV Calculations:

 'Load CCTV_Variables

```

CCTV_CamNo = m_rst("CCTV_CamNo")
CCTV_PerCamW = m_rst("CCTV_PerCamW")
CCTV_PerCamWO = m_rst("CCTV_PerCamWO")
CCTV_IncDur = m_rst("CCTV_IncDur")
CCTV_VMTSav_I = m_rst("CCTV_VMTSav_I")
CCTV_VMTSav_F = m_rst("CCTV_VMTSav_F")

```

 'Traffic and Travel

```

CCTV_PerCamDif = CCTV_PerCamW - CCTV_PerCamWO
CCTV_PerRed = CCTV_IncDur / Avg_IncDur * CCTV_PerCamDif
CCTV_VHTWkSav_BCI = CCTV_PerRed * NonRecVHT_BCI
CCTV_VHTWkSav_BCF = CCTV_PerRed * NonRecVHT_BCF
CCTV_VHTYr_Sav_BCI = CCTV_VHTWkSav_BCI * Ann_Daily_Benefit
CCTV_VHTYr_Sav_BCF = CCTV_VHTWkSav_BCF * Ann_Daily_Benefit
CCTV_FreeInc_BCI = AvgFree_Tot_BCI - ((FreeVMT_BCI + CCTV_VMTSav_I) / (RecFreeVHT_BCI +
NonRecVHT_BCI + CCTV_VHTWkSav_BCI))
CCTV_FreeInc_BCF = AvgFree_Tot_BCF - ((FreeVMT_BCF + CCTV_VMTSav_F) /
(RecFreeVHT_BCF + NonRecVHT_BCF + CCTV_VHTWkSav_BCF))

```

```
CCTV_FreeSpd_BCI = AvgFree_Tot_BCI + CCTV_Freelnc_BCI
CCTV_FreeSpd_BCF = AvgFree_Tot_BCF + CCTV_Freelnc_BCF
```

```
'Travel Time Calculations:
```

```
'Yearly Travel Time Benefits in Minutes (does not consider freight)
```

```
'Assumes no induced ridership
```

```
CCTV_TT_Min_I = CCTV_VHTYr_Sav_BCI * avo * 60
```

```
CCTV_TT_Min_F = CCTV_VHTYr_Sav_BCF * avo * 60
```

```
m_CCTV_TT_Min = (CCTV_TT_Min_F - CCTV_TT_Min_I) * N / 2
```

```
m_CCTV_FrTT_Min = 0
```

```
'Travel Time Benefits in Dollars
```

```
CCTV_TT_Ben_I = (CCTV_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
```

```
CCTV_TT_Ben_F = (CCTV_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
```

```
CCTV_TT_Grad = (CCTV_TT_F - CCTV_TT_I) / N
```

```
m_CCTV_TT_Ben = CCTV_TT_Grad * PofG
```

```
m_CCTV_FrTT_Ben = 0
```

```
'Operating Cost Calculations
```

```
CCTV_VMT_I = CCTV_VMTSav_I * Ann_Daily_Benefit
```

```
CCTV_VMT_F = CCTV_VMTSav_F * Ann_Daily_Benefit
```

```
CCTV_VMT_Tot = (CCTV_VMT_F - CCTV_VMT_I) * N / 2
```

```
'User Cost Benefit Calculations
```

```
If Full_Cost Then
```

```
    CCTV_UserBen_I = (CCTV_VMT_I * Veh_OpCost_Full)
```

```
    CCTV_UserBen_F = (CCTV_VMT_F * Veh_OpCost_Full)
```

```
Else
```

```
    CCTV_UserBen_I = (CCTV_VMT_I * Veh_OpCost_Direct)
```

```
    CCTV_UserBen_F = (CCTV_VMT_F * Veh_OpCost_Direct)
```

```
End If
```

```
CCTV_UCBen_Grad = (CCTV_UserBen_F - CCTV_UserBen_I) / N
```

```
m_CCTV_User_Ben = CCTV_UCBen_Grad * PofG
```

```
m_CCTV_User_Transfer = 0
```

```
'Energy and Emissions
```

```
' (Does not currently differentiate between autos and freight vehicles.)
```

```
'CO Calculations
```

```
CCTV_CORateWO_BCI = m_ikup_Pollution.RetrieveRate(AvgFree_Tot_BCI, em_CO, veht_auto)
```

```
CCTV_CORateW_BCI = m_ikup_Pollution.RetrieveRate(CCTV_FreeSpd_BCI, em_CO, veht_auto)
```

```
CCTV_CORateDif_BCI = CCTV_CORateWO_BCI - CCTV_CORateW_BCI
```

```
CCTV_CORateWO_BCF = m_ikup_Pollution.RetrieveRate(AvgFree_Tot_BCF, em_CO, veht_auto)
```

```
CCTV_CORateW_BCF = m_ikup_Pollution.RetrieveRate(CCTV_FreeSpd_BCF, em_CO, veht_auto)
```

```
CCTV_CORateDif_BCF = CCTV_CORateWO_BCF - CCTV_CORateW_BCF
```

```
CCTV_CO_Gram_BCI = (CCTV_CORateDif_BCI * FreeVMT_BCI) + (CCTV_VMTSav_I * Ann_Daily_Benefit * CO_Rate_Auto)
```

```
CCTV_CO_Gram_BCF = (CCTV_CORateDif_BCF * FreeVMT_BCF) + (CCTV_VMTSav_F * Ann_Daily_Benefit * CO_Rate_Auto)
```

```
CCTV_CO_Gram = (CCTV_CO_Gram_BCI + CCTV_CO_Gram_BCF) / 2 * N
```

```
m_CCTV_CO_Tons = CCTV_CO_Gram / 907184.74
```

```
'NOX Calculations
```

```
CCTV_NOXRateWo_BCI = m_ikup_Pollution.RetrieveRate(AvgFree_Tot_BCI, em_NOX, veht_auto)
```

```
CCTV_NOXRateW_BCI = m_ikup_Pollution.RetrieveRate(CCTV_FreeSpd_BCI, em_NOX, veht_auto)
```

```
CCTV_NOXRateDif_BCI = CCTV_NOXRateWo_BCI - CCTV_NOXRateW_BCI
```

```
CCTV_NOXRateWO_BCF = m_ikup_Pollution.RetrieveRate(AvgFree_Tot_BCF, em_NOX, veht_auto)
```

```
CCTV_NOXRateW_BCF = m_ikup_Pollution.RetrieveRate(CCTV_FreeSpd_BCF, em_NOX, veht_auto)
```

```
CCTV_NOXRateDif_BCF = CCTV_NOXRateWO_BCF - CCTV_NOXRateW_BCF
```

```
CCTV_NOX_Gram_BCI = (CCTV_NOXRateDif_BCI * FreeVMT_BCI) + (CCTV_VMTSav_I * Ann_Daily_Benefit * NOX_Rate_Auto)
```

```

CCTV_NOX_Gram_BCF = (CCTV_NOXRateDif_BCF * FreeVMT_BCF) + (CCTV_VMTsav_F *
Ann_Daily_Benefit * NOX_Rate_Auto)
CCTV_NOX_Gram = (CCTV_NOX_Gram_BCI + CCTV_NOX_Gram_BCF) / 2 * N
m_CCTV_NOX_Tons = CCTV_NOX_Gram / 907184.74

```

'PM10 Calculations

```

CCTV_PM10_Gram_BCI = (CCTV_VMTsav_I * Ann_Daily_Benefit * PM10_Rate_Auto)
CCTV_PM10_Gram_BCF = (CCTV_VMTsav_F * Ann_Daily_Benefit * PM10_Rate_Auto)
CCTV_PM10_Gram = (CCTV_PM10_Gram_BCI + CCTV_PM10_Gram_BCF) / 2 * N
m_CCTV_PM10_Tons = CCTV_PM10_Gram / 907184.74

```

'VOC Calculations

```

CCTV_VOC_Gram_BCI = (CCTV_VMTsav_I * Ann_Daily_Benefit * VOC_Rate_Auto)
CCTV_VOC_Gram_BCF = (CCTV_VMTsav_F * Ann_Daily_Benefit * VOC_Rate_Auto)
CCTV_VOC_Gram = (CCTV_VOC_Gram_BCI + CCTV_VOC_Gram_BCF) / 2 * N
m_CCTV_VOC_Tons = CCTV_VOC_Gram / 907184.74

```

'Emissions Benefit Calculations

```

CCTV_CO_Grad = (CCTV_CO_Gram_BCF - CCTV_CO_Gram_BCI) / N
CCTV_CO_Ben = CCTV_CO_Grad * PofG * COTon_Cost / 907184.74

```

```

CCTV_NOX_Grad = (CCTV_NOX_Gram_BCF - CCTV_NOX_Gram_BCI) / N
CCTV_NOX_Ben = CCTV_NOX_Grad * PofG * NOXTon_Cost / 907184.74

```

```

CCTV_PM10_Grad = (CCTV_PM10_Gram_BCF - CCTV_PM10_Gram_BCI) / N
CCTV_PM10_Ben = CCTV_PM10_Grad * PofG * PM10Ton_Cost / 907184.74

```

```

CCTV_VOC_Grad = (CCTV_VOC_Gram_BCF - CCTV_VOC_Gram_BCI) / N
CCTV_VOC_Ben = CCTV_VOC_Grad * PofG * VOCTon_Cost / 907184.74

```

```

m_CCTV_Env_Ben = CCTV_CO_Ben + CCTV_NOX_Ben + CCTV_PM10_Ben + CCTV_VOC_Ben

```

'SAFETY Calculations

```

CCTV_Fatality_Grad = (CCTV_Fatality_BCF - CCTV_Fatality_BCI) / N
CCTV_Injury_Grad = (CCTV_Injury_BCF - CCTV_Injury_BCI) / N
CCTV_Property_Grad = (CCTV_Property_BCF - CCTV_Property_BCI) / N
m_CCTV_Fatality = CCTV_Fatality_Grad * PofG * Fat_Rate_Auto / 10000000
m_CCTV_Injury = CCTV_Injury_Grad * PofG * Inj_Rate_Auto / 1000000
m_CCTV_Property = CCTV_Property_Grad * PofG * Prop_Rate_Auto / 1000000

```

'Safety Benefit Calculations

```

CCTV_AccBen_Grad = (CCTV_VMT_F - CCTV_VMT_I) / N
CCTV_Fat_Ben = CCTV_AccBen_Grad * PofG * Fatality_Cost * Fat_Rate_Auto / 10000000
CCTV_Inj_Ben = PofG * Evident_Cost * CCTV_AccBen_Grad * Inj_Rate_Auto / 1000000
CCTV_Prop_Ben = PofG * PDO_Cost * CCTV_AccBen_Grad * Prop_Rate_Auto / 1000000
m_CCTV_Safety_Ben = CCTV_Fat_Ben + CCTV_Inj_Ben + CCTV_Prop_Ben

```

'Set module level values (to be displayed)

```

m_Values.Add m_CCTV_TT_Min, "CCTV_TT_Min"
m_Values.Add m_CCTV_FrTT_Min, "CCTV_FrTT_Min"
m_Values.Add m_CCTV_TT_Ben, "CCTV_TT_Ben"
m_Values.Add m_CCTV_FrTT_Ben, "CCTV_FrTT_Ben"
m_Values.Add m_CCTV_User_Ben, "CCTV_User_Ben"
m_Values.Add m_CCTV_User_Transfer, "CCTV_User_Transfer"
m_Values.Add m_CCTV_CO_Tons, "CCTV_CO_Tons"
m_Values.Add m_CCTV_NOX_Tons, "CCTV_NOX_Tons"
m_Values.Add m_CCTV_PM10_Tons, "CCTV_PM10_Tons"
m_Values.Add m_CCTV_VOC_Tons, "CCTV_Voc_Tons"
m_Values.Add m_CCTV_Env_Ben, "CCTV_Env_Ben"
m_Values.Add m_CCTV_Fatality, "CCTV_Fatality"
m_Values.Add m_CCTV_Injury, "CCTV_Injury"

```

```

    m_Values.Add m_CCTV_Property, "CCTV_Property"
    m_Values.Add m_CCTV_Safety_Ben, "CCTV_Safety_Ben"

'Calls local function to set cost nodes
CalculateCosts "CCTV", Discount_Rate, N

End If
End If

If bDET Then
    Call SetSCRITSType("its_calc_SCRITS_DET")
    If Not m_rst.EOF Then

        '-----
        'Traffic Detection And Information:

        'Load Traffic Detection And Information Variables
        DET_PerCamW = m_rst("DET_PerW")
        DET_PerCamWO = m_rst("DET_PerWO")
        Det_IncDur = m_rst("DET_IncDur")
        Det_VMTSav_I = m_rst("DET_VMTSav_I")
        Det_VMTSav_F = m_rst("DET_VMTSav_F")

        'Traffic and Travel
        Det_PerCamDif = DET_PerCamW - DET_PerCamWO
        Det_PerREd = Det_IncDur / Avg_IncDur * Det_PerCamDif
        DET_VHTWkSav_BCI = Det_PerREd * NonRecVHT_BCI
        DET_VHTWkSav_BCF = Det_PerREd * NonRecVHT_BCF
        DET_VHTYr_Sav_BCI = DET_VHTWkSav_BCI * Ann_Daily_Benefit
        DET_VHTYr_Sav_BCF = DET_VHTWkSav_BCF * Ann_Daily_Benefit
        Det_FreeInc_BCI = AvgFree_Tot_BCI - ((FreeVMT_BCI + Det_VMTSav_I) / (RecFreeVHT_BCI +
NonRecVHT_BCI + Det_VHT_WkSav))
        Det_FreeInc_BCF = AvgFree_Tot_BCF - ((FreeVMT_BCF + Det_VMTSav_F) / (RecFreeVHT_BCF +
NonRecVHT_BCF + Det_VHT_WkSav))
        DET_FreeSpd_BCI = AvgFree_Tot_BCI + Det_FreeInc_BCI
        DET_FreeSpd_BCF = AvgFree_Tot_BCF + Det_FreeInc_BCF

        'Travel Time Calculations:
        'Yearly Travel Time Benefits in Minutes (does not consider freight)
        'Assumes no induced ridership
        DET_TT_Min_I = DET_VHTYr_Sav_BCI * avo * 60
        DET_TT_Min_F = DET_VHTYr_Sav_BCF * avo * 60
        m_DET_TT_Min = (DET_TT_Min_F - DET_TT_Min_I) * N / 2
        m_DET_FrTT_Min = 0

        'Travel Time Benefits in Dollars
        DET_TT_Ben_I = (DET_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
        DET_TT_Ben_F = (DET_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)

        DET_TT_Ben_Grad = (DET_TT_Ben_F - DET_TT_Ben_I) / N
        DET_TT_Ben = DET_TT_Ben_Grad * PofG
        DET_FrTT_Ben = 0

        'Operating Cost Calculations
        Det_VMT_I = Det_VMTSav_I * Ann_Daily_Benefit
        Det_VMT_F = Det_VMTSav_F * Ann_Daily_Benefit
        DET_VMT_Grad = (DET_VMT_F - Det_VMT_I) / N

        'User Cost Benefit Calculations
        If Full_Cost Then
            DET_UserBen_I = (Det_VMT_I * Veh_OpCost_Full)
            DET_UserBen_F = (DET_VMT_F * Veh_OpCost_Full)

```

```

Else
  DET_UserBen_I = (Det_VMT_I * Veh_OpCost_Direct)
  DET_UserBen_F = (DET_VMT_F * Veh_OpCost_Direct)
End If

DET_User_Ben_Grad = (DET_UserBen_F - DET_UserBen_I) / N
m_DET_User_Ben = DET_User_Ben_Grad * PofG

'Energy and Emissions
'(Does not currently differentiate between autos and freight vehicles.)
'CO Calculations
DET_CORateWO_BCI = m_Ikup_Pollution.RetrieveRate(AvgFree_Tot_BCI, em_CO, veht_auto)
DET_CORateW_BCI = m_Ikup_Pollution.RetrieveRate(DET_FreeSpd_BCI, em_CO, veht_auto)
DET_CORateDif_BCI = DET_CORateW_BCI - DET_CORateWO_BCI
DET_CORateWO_BCF = m_Ikup_Pollution.RetrieveRate(AvgFree_Tot_BCF, em_CO, veht_auto)
DET_CORateW_BCF = m_Ikup_Pollution.RetrieveRate(DET_FreeSpd_BCF, em_CO, veht_auto)
DET_CORateDif_BCF = DET_CORateW_BCF - DET_CORateWO_BCF
DET_CO_Gram_BCI = (DET_CORateDif_BCI * FreeVMT_BCI) + (Det_VMTSav_I * Ann_Daily_Benefit
* CO_Rate_Auto)
DET_CO_Gram_BCF = (DET_CORateDif_BCF * FreeVMT_BCF) + (Det_VMTSav_F *
Ann_Daily_Benefit * CO_Rate_Auto)
DET_CO_Gram = (DET_CO_Gram_BCI + DET_CO_Gram_BCF) / 2 * N
m_DET_CO_Tons = DET_CO_Gram / 907184.74

'NOX Calculations
DET_NOXRateWO_BCI = m_Ikup_Pollution.RetrieveRate(AvgFree_Tot_BCI, em_NOX, veht_auto)
DET_NOXRateW_BCI = m_Ikup_Pollution.RetrieveRate(DET_FreeSpd_BCI, em_NOX, veht_auto)
DET_NOXRateDif_BCI = DET_NOXRateW_BCI - DET_NOXRateWO_BCI
DET_NOXRateWO_BCF = m_Ikup_Pollution.RetrieveRate(AvgFree_Tot_BCF, em_NOX, veht_auto)
DET_NOXRateW_BCF = m_Ikup_Pollution.RetrieveRate(DET_FreeSpd_BCF, em_NOX, veht_auto)
DET_NOXRateDif_BCF = DET_NOXRateW_BCF - DET_NOXRateWO_BCF
DET_NOX_Gram_BCI = (DET_NOXRateDif_BCI * FreeVMT_BCI) + (Det_VMTSav_I *
Ann_Daily_Benefit * NOX_Rate_Auto)
DET_NOX_Gram_BCF = (DET_NOXRateDif_BCF * FreeVMT_BCF) + (Det_VMTSav_F *
Ann_Daily_Benefit * NOX_Rate_Auto)
DET_NOX_Gram = (DET_NOX_Gram_BCI + DET_NOX_Gram_BCF) / 2 * N
m_DET_NOX_Tons = DET_NOX_Gram / 907184.74

'PM10 Calculations
DET_PM10_Gram_BCI = (Det_VMTSav_I * Ann_Daily_Benefit * PM10_Rate_Auto)
DET_PM10_Gram_BCF = (Det_VMTSav_F * Ann_Daily_Benefit * PM10_Rate_Auto)
DET_PM10_Gram = (DET_PM10_Gram_BCI + DET_PM10_Gram_BCF) / 2 * N
m_DET_PM10_Tons = DET_PM10_Gram / 907184.74

'VOC Calculations
DET_VOC_Gram_BCI = (Det_VMTSav_I * Ann_Daily_Benefit * VOC_Rate_Auto)
DET_VOC_Gram_BCF = (Det_VMTSav_F * Ann_Daily_Benefit * VOC_Rate_Auto)
DET_VOC_Gram = (DET_VOC_Gram_BCI + DET_VOC_Gram_BCF) / 2 * N
m_DET_VOC_Tons = DET_VOC_Gram / 907184.74

'Emissions Benefit Calculations
DET_CO_Grad = (DET_CO_Gram_BCF - DET_CO_Gram_BCI) / N
DET_CO_Ben = DET_CO_Grad * PofG * COTon_Cost / 907184.74

DET_NOX_Grad = (DET_NOX_Gram_BCF - DET_NOX_Gram_BCI) / N
DET_NOX_Ben = DET_NOX_Grad * PofG * NOXTon_Cost / 907184.74

DET_PM10_Grad = (DET_PM10_Gram_BCF - DET_PM10_Gram_BCI) / N
DET_PM10_Ben = DET_PM10_Grad * PofG * PM10Ton_Cost / 907184.74

DET_VOC_Grad = (DET_VOC_Gram_BCF - DET_VOC_Gram_BCI) / N

```

```
DET_VOC_Ben = DET_VOC_Grad * PofG * VOCTon_Cost / 907184.74
```

```
m_DET_Env_Ben = DET_CO_Ben + DET_NOX_Ben + DET_PM10_Ben + DET_VOC_Ben
```

```
'SAFETY Calculations
```

```
m_DET_Fatality = DET_VMT_Grad * N / 2 * Fat_Rate_Auto / 100000000
```

```
m_DET_Injury = DET_VMT_Grad * N / 2 * Inj_Rate_Auto / 1000000
```

```
m_DET_Property = DET_VMT_Grad * N / 2 * Prop_Rate_Auto / 1000000
```

```
'Safety Benefit Calculations
```

```
DET_VMT_Grad = (DET_VMT_F - Det_VMT_I) / N
```

```
DET_Fat_Ben = DET_VMT_Grad * PofG * Fatality_Cost * Fat_Rate_Auto / 100000000
```

```
DET_Inj_Ben = DET_VMT_Grad * PofG * Evident_Cost * Inj_Rate_Auto / 1000000
```

```
DET_Prop_Ben = DET_VMT_Grad * PofG * PDO_Cost * Prop_Rate_Auto / 1000000
```

```
m_DET_Safety_Ben = DET_Fat_Ben + DET_Inj_Ben + DET_Prop_Ben
```

```
'Set module level values (to be displayed)
```

```
  m_Values.Add m_DET_TT_Min, "DET_TT_Min"
```

```
  m_Values.Add m_DET_FrTT_Min, "DET_FrTT_Min"
```

```
  m_Values.Add m_DET_TT_Ben, "DET_TT_Ben"
```

```
  m_Values.Add m_DET_FrTT_Ben, "DET_FrTT_Ben"
```

```
  m_Values.Add m_DET_CO_Tons, "DET_CO_Tons"
```

```
  m_Values.Add m_DET_NOX_Tons, "DET_NOX_Tons"
```

```
  m_Values.Add m_DET_PM10_Tons, "DET_PM10_Tons"
```

```
  m_Values.Add m_DET_VOC_Tons, "DET_Voc_Tons"
```

```
  m_Values.Add m_DET_Env_Ben, "DET_Env_Ben"
```

```
  m_Values.Add m_DET_User_Ben, "DET_User_Ben"
```

```
  m_Values.Add m_DET_User_Transfer, "DET_User_Transfer"
```

```
  m_Values.Add m_DET_Fatality, "DET_Fatality"
```

```
  m_Values.Add m_DET_Injury, "DET_Injury"
```

```
  m_Values.Add m_DET_Property, "DET_Property"
```

```
  m_Values.Add m_DET_Safety_Ben, "DET_Safety_Ben"
```

```
'Calls local function to set cost nodes
```

```
CalculateCosts "DET", Discount_Rate, N
```

```
End If
```

```
End If
```

```
If bHAR Then
```

```
  Call SetSCRITSType("its_calc_SCRITS_HAR")
```

```
  If Not m_rst.EOF Then
```

```
    '-----
```

```
    'Highway Advisory Radio
```

```
    'Traffic and Travel
```

```
    'Load Highway Advisory Radio Variables
```

```
    HAR_No = m_rst("HAR_No")
```

```
    HAR_Vol_I = m_rst("HAR_Vol_I")
```

```
    HAR_Vol_F = m_rst("HAR_Vol_F")
```

```
    HAR_Time = m_rst("HAR_Time")
```

```
    HAR_Act = m_rst("HAR_Act")
```

```
    HAR_Tuneper = m_rst("HAR_Tuneper")
```

```
    HAR_Saveper = m_rst("HAR_Saveper")
```

```
    HAR_SaveTime = m_rst("HAR_SaveTime")
```

```

Har_VHTSavUn_I = HAR_Vol_I * HAR_Time * HAR_Act * HAR_Tuneper * HAR_Saveper *
HAR_SaveTime / 60
Har_VHTSavUn_F = HAR_Vol_F * HAR_Time * HAR_Act * HAR_Tuneper * HAR_Saveper *
HAR_SaveTime / 60
Har_VHTDySav_I = Har_VHTSavUn_I * HAR_No
Har_VHTDySav_F = Har_VHTSavUn_F * HAR_No
Har_PerNonSav_I = Har_VHTDySav_I / NonRecVHT_BCI
Har_PerNonSav_F = Har_VHTDySav_F / NonRecVHT_BCF
Har_VHTYrSav_I = Har_VHTDySav_I * Ann_Daily_Benefit
Har_VHTYrSav_F = Har_VHTDySav_F * Ann_Daily_Benefit

```

'Travel Time Calculations:

'Yearly Travel Time Benefits in Minutes (does not consider freight)

'Assumes no induced ridership

HAR_TT_Min_I = Har_VHTYrSav_I * avo * 60

HAR_TT_Min_F = Har_VHTYrSav_F * avo * 60

m_HAR_TT_Min = (HAR_TT_Min_F - HAR_TT_Min_I) * N / 2

m_HAR_FrTT_Min = 0

'Travel Time Benefits in Dollars

HAR_TT_Ben_I = (HAR_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)

HAR_TT_Ben_F = (HAR_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)

HAR_TTBen_Grad = (HAR_TT_Ben_F - HAR_TT_Ben_I) / N

m_HAR_TT_Ben = HAR_TTBen_Grad * PofG

m_HAR_FrTT_Ben = 0

'Operating Cost Calculations SCRITS assumes no VMT change for HAR projects

m_HAR_User_Ben = 0

m_HAR_User_Transfer = 0

'Energy and Emissions

' (SCRITS assumes energy and emissions changes negligible for HAR)

m_HAR_CO_Tons = 0

m_HAR_NOX_Tons = 0

m_HAR_PM10_Tons = 0

m_HAR_VOC_Tons = 0

m_HAR_Env_Ben = 0

'SAFETY Calculations

Har_SecFatDy_I = FreeVMT_BCI * SecAccPer * Fat_Rate_Auto / 100000000

Har_SecFatDy_F = FreeVMT_BCF * SecAccPer * Fat_Rate_Auto / 100000000

Har_FatRedDy_I = Har_PerNonSav_I * Har_SecFatDy_I

Har_FatRedDy_F = Har_PerNonSav_F * Har_SecFatDy_F

HAR_Fatality_I = Har_FatRedDy_I * Ann_Daily_Benefit

HAR_Fatality_F = Har_FatRedDy_F * Ann_Daily_Benefit

Har_SecInjDy_I = FreeVMT_BCI * SecAccPer * Inj_Rate_Auto / 100000000

Har_SecInjDy_F = FreeVMT_BCF * SecAccPer * Inj_Rate_Auto / 100000000

Har_InjRedDy_I = Har_PerNonSav_I * Har_SecInjDy_I

Har_InjRedDy_F = Har_PerNonSav_F * Har_SecInjDy_F

HAR_Injury_I = Har_InjRedDy_I * Ann_Daily_Benefit

HAR_Injury_F = Har_InjRedDy_F * Ann_Daily_Benefit

Har_SecPropDy_I = FreeVMT_BCI * SecAccPer * Prop_Rate_Auto / 100000000

Har_SecPropDy_F = FreeVMT_BCF * SecAccPer * Prop_Rate_Auto / 100000000

Har_PropRedDy_I = Har_PerNonSav_I * Har_SecPropDy_I

Har_PropRedDy_F = Har_PerNonSav_F * Har_SecPropDy_F

HAR_Property_I = Har_PropRedDy_I * Ann_Daily_Benefit

HAR_Property_F = Har_PropRedDy_F * Ann_Daily_Benefit

HAR_Fatality_Grad = (HAR_Fatality_F - HAR_Fatality_I) / N

m_HAR_Fatality = HAR_Fatality_Grad * N / 2

HAR_Injury_Grad = (HAR_Injury_F - HAR_Injury_I) / N

m_HAR_Injury = HAR_Injury_Grad * N / 2

HAR_Property_Grad = (HAR_Property_F - HAR_Property_I) / N


```
m_HAR_Property = HAR_Property_Grad * N / 2
```

```
'Safety Benefit Calculations
```

```
Har_Fat_Ben = HAR_Fatality_Grad * PofG * Fatality_Cost
```

```
Har_Inj_Ben = HAR_Injury_Grad * PofG * Evident_Cost
```

```
Har_Prop_Ben = HAR_Property_Grad * PofG * PDO_Cost
```

```
m_HAR_Safety_Ben = Har_Fat_Ben + Har_Inj_Ben + Har_Prop_Ben
```

```
'Set module level values (to be displayed)
```

```
  m_Values.Add m_HAR_TT_Min, "Har_TT_Min"
```

```
  m_Values.Add m_HAR_FrTT_Min, "Har_FrTT_Min"
```

```
  m_Values.Add m_HAR_TT_Ben, "Har_TT_Ben"
```

```
  m_Values.Add m_HAR_FrTT_Ben, "Har_FrTT_Ben"
```

```
  m_Values.Add m_HAR_CO_Tons, "Har_CO_Tons"
```

```
  m_Values.Add m_HAR_NOX_Tons, "Har_NOX_Tons"
```

```
  m_Values.Add m_HAR_PM10_Tons, "Har_PM10_Tons"
```

```
  m_Values.Add m_HAR_VOC_Tons, "Har_Voc_Tons"
```

```
  m_Values.Add m_HAR_Env_Ben, "Har_Env_Ben"
```

```
  m_Values.Add m_HAR_User_Ben, "Har_User_Ben"
```

```
  m_Values.Add m_HAR_User_Transfer, "Har_User_Transfer"
```

```
  m_Values.Add m_HAR_Fatality, "Har_Fatality"
```

```
  m_Values.Add m_HAR_Injury, "Har_Injury"
```

```
  m_Values.Add m_HAR_Property, "Har_Property"
```

```
  m_Values.Add m_HAR_Safety_Ben, "Har_Safety_Ben"
```

```
'Calls local function to set cost nodes
```

```
CalculateCosts "HAR", Discount_Rate, N
```

```
End If
```

```
End If
```

```
If bVMS Then
```

```
  Call SetSCRITSType("its_calc_SCRITS_VMS")
```

```
  If Not m_rst.EOF Then
```

```
    '-----
```

```
    'Variable Message Sign
```

```
    'Load Variable Message Sign Variables
```

```
    VMS_No = m_rst("VMS_No")
```

```
    VMS_Vol_I = m_rst("VMS_Vol_I")
```

```
    VMS_Vol_F = m_rst("VMS_Vol_F")
```

```
    VMS_Hrs = m_rst("VMS_Hrs")
```

```
    VMS_Act = m_rst("VMS_Act")
```

```
    VMS_Saveper = m_rst("VMS_Saveper")
```

```
    VMS_SaveTime = m_rst("VMS_SaveTime")
```

```
    'Traffic and Travel
```

```
    VMS_VHTSavUn_I = VMS_Vol_I * VMS_Hrs * VMS_Act * VMS_Saveper * VMS_SaveTime / 60
```

```
    VMS_VHTSavUn_F = VMS_Vol_F * VMS_Hrs * VMS_Act * VMS_Saveper * VMS_SaveTime / 60
```

```
    VMS_VHTDySav_I = VMS_VHTSavUn_I * VMS_No
```

```
    VMS_VHTDySav_F = VMS_VHTSavUn_F * VMS_No
```

```
    VMS_PerNonSav_I = VMS_VHTDySav_I / NonRecVHT_BCI
```

```
    VMS_PerNonSav_F = VMS_VHTDySav_F / NonRecVHT_BCF
```

```
    VMS_VHTYrSav_I = VMS_VHTDySav_I * Ann_Daily_Benefit
```

```
    VMS_VHTYrSav_F = VMS_VHTDySav_F * Ann_Daily_Benefit
```

'Travel Time Calculations:

'Yearly Travel Time Benefits in Minutes (does not consider freight)

'Assumes no induced ridership

VMS_TT_Min_I = VMS_VHTYrSav_I * avo * 60

VMS_TT_Min_F = VMS_VHTYrSav_F * avo * 60

m_VMS_TT_Min = (VMS_TT_Min_F - VMS_TT_Min_I) * N / 2

m_VMS_FrTT_Min = 0

'Travel Time Benefits in Dollars

VMS_TT_Ben_I = (VMS_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)

VMS_TT_Ben_F = (VMS_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)

VMS_TTBen_Grad = (VMS_TT_Ben_F - VMS_TT_Ben_I) / N

m_VMS_TT_Ben = PofG * VMS_TTBen_Grad

m_VMS_FrTT_Ben = 0

'Operating Cost Calculations

'SCRITS assumes no VMT change for VMS projects

m_VMS_User_Ben = 0

m_VMS_User_Transfer = 0

'Energy and Emissions

' (SCRITS assumes energy and emissions changes negligible for VMS)

m_VMS_CO_Tons = 0

m_VMS_NOX_Tons = 0

m_VMS_PM10_Tons = 0

m_VMS_VOC_Tons = 0

m_VMS_Env_Ben = 0

'SAFETY Calculations

VMS_SecFatDy_I = FreeVMT_BCI * SecAccPer * Fat_Rate_Auto / 100000000

VMS_SecFatDy_F = FreeVMT_BCF * SecAccPer * Fat_Rate_Auto / 100000000

VMS_FatRedDy_I = VMS_PerNonSav_I * VMS_SecFatDy_I

VMS_FatRedDy_F = VMS_PerNonSav_F * VMS_SecFatDy_F

VMS_Fatality_I = VMS_FatRedDy_I * Ann_Daily_Benefit

VMS_Fatality_F = VMS_FatRedDy_F * Ann_Daily_Benefit

VMS_SecInjDy_I = FreeVMT_BCI * SecAccPer * Inj_Rate_Auto / 100000000

VMS_SecInjDy_F = FreeVMT_BCF * SecAccPer * Inj_Rate_Auto / 100000000

VMS_InjRedDy_I = VMS_PerNonSav_I * VMS_SecInjDy_I

VMS_InjRedDy_F = VMS_PerNonSav_F * VMS_SecInjDy_F

VMS_Injury_I = VMS_InjRedDy_I * Ann_Daily_Benefit

VMS_Injury_F = VMS_InjRedDy_F * Ann_Daily_Benefit

VMS_SecPropDy_I = FreeVMT_BCI * SecAccPer * Prop_Rate_Auto / 100000000

VMS_SecPropDy_F = FreeVMT_BCF * SecAccPer * Prop_Rate_Auto / 100000000

VMS_PropRedDy_I = VMS_PerNonSav_I * VMS_SecPropDy_I

VMS_PropRedDy_F = VMS_PerNonSav_F * VMS_SecPropDy_F

VMS_Property_I = VMS_PropRedDy_I * Ann_Daily_Benefit

VMS_Property_F = VMS_PropRedDy_F * Ann_Daily_Benefit

m_VMS_Fatality = (VMS_Fatality_F - VMS_Fatality_I) * N / 2

m_VMS_Injury = (VMS_Injury_F - VMS_Injury_I) * N / 2

m_VMS_Property = (VMS_Property_F - VMS_Property_I) * N / 2

'Safety Benefit Calculations

VMS_Fatality_Grad = (VMS_Fatality_F - VMS_Fatality_I) / N

VMS_Fat_Ben = VMS_Fatality_Grad * PofG * Fatality_Cost

VMS_Injury_Grad = (VMS_Injury_F - VMS_Injury_I) / N

VMS_Inj_Ben = VMS_Injury_Grad * PofG * Evident_Cost

VMS_Property_Grad = (VMS_Property_F - VMS_Property_I) / N

VMS_Prop_Ben = VMS_Property_Grad * PofG * PDO_Cost

m_VMS_Safety_Ben = VMS_Fat_Ben + VMS_Inj_Ben + VMS_Prop_Ben

'Set module level values (to be displayed)

```

m_Values.Add m_VMS_TT_Min, "VMS_TT_Min"
m_Values.Add m_VMS_FrTT_Min, "VMS_FrTT_Min"
m_Values.Add m_VMS_TT_Ben, "VMS_TT_Ben"
m_Values.Add m_VMS_FrTT_Ben, "VMS_FrTT_Ben"

m_Values.Add m_VMS_CO_Tons, "VMS_CO_Tons"
m_Values.Add m_VMS_NOX_Tons, "VMS_NOX_Tons"
m_Values.Add m_VMS_PM10_Tons, "VMS_PM10_Tons"
m_Values.Add m_VMS_VOC_Tons, "VMS_Voc_Tons"
m_Values.Add m_VMS_Env_Ben, "VMS_Env_Ben"

m_Values.Add m_VMS_User_Ben, "VMS_User_Ben"
m_Values.Add m_VMS_User_Transfer, "VMS_User_Transfer"

m_Values.Add m_VMS_Fatality, "VMS_Fatality"
m_Values.Add m_VMS_Injury, "VMS_Injury"
m_Values.Add m_VMS_Property, "VMS_Property"
m_Values.Add m_VMS_Safety_Ben, "VMS_Safety_Ben"

'Calls local function to set cost nodes
CalculateCosts "VMS", Discount_Rate, N

End If
End If

If bATIS Then
Call SetSCRITSType("its_calc_SCRITS_Pag")
If Not m_rst.EOF Then

'-----
'Pager Based ATIS

'Load Variable Message Sign Variables
Pag_TripPer = m_rst("Pag_TripPer")
Pag_SysPer = m_rst("PAG_SysPer")
Pag_ActPer = m_rst("PAG_ActPer")
Pag_SavePer = m_rst("PAG_SavPer")
Pag_SaveMin = m_rst("PAG_SavMin")

'Traffic and Travel
Avg_TripHr_I = AvgTripNo_BCI * Avg_TripMin_BCI / 60
Avg_TripHr_F = AvgTripNo_BCF * Avg_TripMin_BCF / 60
PAG_DySav_I = AvgTripNo_BCI * Pag_TripPer * Pag_SysPer * Pag_ActPer * Pag_SavePer *
Pag_SaveMin / 60
PAG_DySav_F = AvgTripNo_BCF * Pag_TripPer * Pag_SysPer * Pag_ActPer * Pag_SavePer *
Pag_SaveMin / 60
PAG_PerSav_I = PAG_DySav_I / Avg_TripHr_I
PAG_PerSav_F = PAG_DySav_F / Avg_TripHr_F
PAG_YrSav_I = PAG_DySav_I * Ann_Daily_Benefit
PAG_YrSav_F = PAG_DySav_F * Ann_Daily_Benefit

'Travel Time Calculations:
'Yearly Travel Time Benefits in Minutes (does not consider freight)
'Assumes no induced travel
PAG_TT_Min_I = PAG_YrSav_I * avo * 60
PAG_TT_Min_F = PAG_YrSav_F * avo * 60
m_PAG_TT_Min = (PAG_TT_Min_F - PAG_TT_Min_I) * N / 2
m_PAG_FrTT_Min = 0

'Travel Time Benefits in Dollars
PAG_TT_Ben_I = (PAG_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)

```

```

PAG_TT_Ben_F = (PAG_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
PAG_TTBen_Grad = (PAG_TT_Min_F - PAG_TT_Min_I) / N
m_PAG_TT_Ben = PAG_TTBen_Grad * PofG
m_PAG_FrTT_Ben = 0

```

```

'Operating Cost Calculations SCRITS assumes no VMT change for PAG projects
m_PAG_User_Ben = 0
m_PAG_User_Transfer = 0

```

```

'Energy and Emissions
'(SCRITS assumes energy and emissions changes negligible for PAG)
m_PAG_CO_Tons = 0
m_PAG_NOX_Tons = 0
m_PAG_PM10_Tons = 0
m_PAG_VOC_Tons = 0
m_PAG_Env_Ben = 0

```

```

'SAFETY Calculations
PAG_SecFatDy_I = FreeVMT_BCI * SecAccPer * Fat_Rate_Auto / 100000000
PAG_SecFatDy_F = FreeVMT_BCF * SecAccPer * Fat_Rate_Auto / 100000000
PAG_FatRedDy_I = PAG_PerSav_I * PAG_SecFatDy_I
PAG_FatRedDy_F = PAG_PerSav_F * PAG_SecFatDy_F
PAG_Fatality_I = PAG_FatRedDy_I * Ann_Daily_Benefit
PAG_Fatality_F = PAG_FatRedDy_F * Ann_Daily_Benefit
PAG_SecInjDy_I = FreeVMT_BCI * SecAccPer * Inj_Rate_Auto / 100000000
PAG_SecInjDy_F = FreeVMT_BCF * SecAccPer * Inj_Rate_Auto / 100000000
PAG_InjRedDy_I = PAG_PerSav_I * PAG_SecInjDy_I
PAG_InjRedDy_F = PAG_PerSav_F * PAG_SecInjDy_F
PAG_Injury_I = PAG_InjRedDy_I * Ann_Daily_Benefit
PAG_Injury_F = PAG_InjRedDy_F * Ann_Daily_Benefit
PAG_SecPropDy_I = FreeVMT_BCI * SecAccPer * Prop_Rate_Auto / 100000000
PAG_SecPropDy_F = FreeVMT_BCF * SecAccPer * Prop_Rate_Auto / 100000000
PAG_PropRedDy_I = PAG_PerSav_I * PAG_SecPropDy_I
PAG_PropRedDy_F = PAG_PerSav_F * PAG_SecPropDy_F
PAG_Property_I = PAG_PropRedDy_I * Ann_Daily_Benefit
PAG_Property_F = PAG_PropRedDy_F * Ann_Daily_Benefit
m_PAG_Fatality = (PAG_Fatality_F - PAG_Fatality_I) * N / 2
m_PAG_Injury = (PAG_Injury_F - PAG_Injury_I) * N / 2
m_PAG_Property = (PAG_Property_F - PAG_Property_I) * N / 2

```

```

'Safety Benefit Calculations
PAG_Fatality_Grad = (PAG_Fatality_F - PAG_Fatality_I) / N
PAG_Fat_Ben = PAG_Fatality_Grad * PofG * Fatality_Cost
PAG_Injury_Grad = (PAG_Injury_F - PAG_Injury_I) / N
PAG_Inj_Ben = PAG_Injury_Grad * PofG * Evident_Cost
PAG_Property_Grad = (PAG_Property_F - PAG_Property_I) / N
PAG_Prop_Ben = PAG_Property_Grad * PofG * PDO_Cost
m_PAG_Safety_Ben = PAG_Fat_Ben + PAG_Inj_Ben + PAG_Prop_Ben

```

```

'Set module level values (to be displayed)
m_Values.Add m_PAG_TT_Min, "PAG_TT_Min"
m_Values.Add m_PAG_FrTT_Min, "PAG_FrTT_Min"
m_Values.Add m_PAG_TT_Ben, "PAG_TT_Ben"
m_Values.Add m_PAG_FrTT_Ben, "PAG_FrTT_Ben"

m_Values.Add m_PAG_CO_Tons, "PAG_CO_Tons"
m_Values.Add m_PAG_NOX_Tons, "PAG_NOX_Tons"
m_Values.Add m_PAG_PM10_Tons, "PAG_PM10_Tons"
m_Values.Add m_PAG_VOC_Tons, "PAG_Voc_Tons"
m_Values.Add m_PAG_Env_Ben, "PAG_Env_Ben"

m_Values.Add m_PAG_User_Ben, "PAG_User_Ben"

```

```

    m_Values.Add m_PAG_User_Transfer, "PAG_User_Transfer"

    m_Values.Add m_PAG_Fatality, "PAG_Fatality"
    m_Values.Add m_PAG_Injury, "PAG_Injury"
    m_Values.Add m_PAG_Property, "PAG_Property"
    m_Values.Add m_PAG_Safety_Ben, "PAG_Safety_Ben"

    'Calls local function to set cost nodes
    CalculateCosts "PAG", Discount_Rate, N

    End If
End If

If bKio Then
    Call SetSCRITSType("its_calc_SCRITS_Kio")
    If Not m_rst.EOF Then

        '-----
        'Traffic Information Kiosks

        'Load Variable Message Sign Variables
        Kio_No = m_rst("KIO_No")
        Kio_TripPer_I = m_rst("KIO_TripPer_I")
        Kio_TripPer_F = m_rst("KIO_TripPer_F")
        Kio_Look = m_rst("KIO_Look")
        Kio_SavePer = m_rst("KIO_SavPer")
        Kio_SaveMin = m_rst("KIO_SavMin")

        'Traffic and Travel
        Avg_TripHr_I = AvgTripNo_BCI * Avg_TripMin_BCI / 60
        Avg_TripHr_F = AvgTripNo_BCF * Avg_TripMin_BCF / 60
        Kio_TripTot_I = Kio_No * Kio_TripPer_I
        Kio_TripTot_F = Kio_No * Kio_TripPer_F
        KIO_DySav_I = Kio_TripTot_I * Kio_Look * Kio_SavePer * Kio_SaveMin / 60
        KIO_DySav_F = Kio_TripTot_F * Kio_Look * Kio_SavePer * Kio_SaveMin / 60
        KIO_PerSav_I = KIO_DySav_I / Avg_TripHr_I
        KIO_PerSav_F = KIO_DySav_F / Avg_TripHr_F
        KIO_YrSav_I = KIO_DySav_I * Ann_Daily_Benefit
        KIO_YrSav_F = KIO_DySav_F * Ann_Daily_Benefit

        'Travel Time Calculations:
        'Yearly Travel Time Benefits in Minutes (freight considered in separate freight kiosk category)
        'Assumes no induced travel
        KIO_TT_Min_I = KIO_YrSav_I * avo * 60
        KIO_TT_Min_F = KIO_YrSav_F * avo * 60
        m_KIO_TT_Min = (KIO_TT_Min_F - KIO_TT_Min_I) * N / 2
        m_KIO_FrTT_Min = 0

        'Travel Time Benefits in Dollars
        KIO_TT_Ben_I = (KIO_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
        KIO_TT_Ben_F = (KIO_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
        KIO_TTBen_Grad = (KIO_TT_Ben_F - KIO_TT_Ben_I) / N
        m_KIO_TT_Ben = KIO_TTBen_Grad * PofG
        m_KIO_FrTT_Ben = 0

        'Operating Cost Calculations
        'SCRITS assumes no VMT change for KIO projects
        m_KIO_User_Ben = 0
        m_KIO_User_Transfer = 0

        'Energy and Emissions
        '(SCRITS assumes energy and emissions changes negligible for KIO)

```

```

m_KIO_CO_Tons = 0
m_KIO_NOX_Tons = 0
m_KIO_PM10_Tons = 0
m_KIO_VOC_Tons = 0
m_KIO_Env_Ben = 0

```

'SAFETY Calculations

```

KIO_SecFatDy_I = FreeVMT_BCI * SecAccPer * Fat_Rate_Auto / 100000000
KIO_SecFatDy_F = FreeVMT_BCF * SecAccPer * Fat_Rate_Auto / 100000000
KIO_FatRedDy_I = KIO_PerSav_I * KIO_SecFatDy_I
KIO_FatRedDy_F = KIO_PerSav_F * KIO_SecFatDy_I
KIO_Fatality_I = KIO_FatRedDy_I * Ann_Daily_Benefit
KIO_Fatality_F = KIO_FatRedDy_F * Ann_Daily_Benefit
KIO_SecInjDy_I = FreeVMT_BCI * SecAccPer * Inj_Rate_Auto / 100000000
KIO_SecInjDy_F = FreeVMT_BCF * SecAccPer * Inj_Rate_Auto / 100000000
KIO_InjRedDy_I = KIO_PerSav_I * KIO_SecInjDy_I
KIO_InjRedDy_F = KIO_PerSav_F * KIO_SecInjDy_I
KIO_Injury_I = KIO_InjRedDy_I * Ann_Daily_Benefit
KIO_Injury_F = KIO_InjRedDy_F * Ann_Daily_Benefit
KIO_SecPropDy_I = FreeVMT_BCI * SecAccPer * Prop_Rate_Auto / 100000000
KIO_SecPropDy_F = FreeVMT_BCF * SecAccPer * Prop_Rate_Auto / 100000000
KIO_PropRedDy_I = KIO_PerSav_I * KIO_SecPropDy_I
KIO_PropRedDy_F = KIO_PerSav_F * KIO_SecPropDy_I
KIO_Property_I = KIO_PropRedDy_I * Ann_Daily_Benefit
KIO_Property_F = KIO_PropRedDy_F * Ann_Daily_Benefit

```

```

m_KIO_Fatality = (KIO_Fatality_F - KIO_Fatality_I) * N / 2
m_KIO_Injury = (KIO_Injury_F - KIO_Injury_I) * N / 2
m_KIO_Property = (KIO_Property_F - KIO_Property_I) * N / 2

```

'Safety Benefit Calculations

```

KIO_FatBen_Grad = (KIO_Fatality_F - KIO_Fatality_I) / N
KIO_Fat_Ben = PofG * Fatality_Cost * KIO_FatBen_Grad
KIO_InjBen_Grad = (KIO_Injury_F - KIO_Injury_I) / N
KIO_Inj_Ben = PofG * Evident_Cost * KIO_InjBen_Grad
KIO_PropBen_Grad = (KIO_Property_F - KIO_Property_I) / N
KIO_Prop_Ben = PofG * PDO_Cost * KIO_PropBen_Grad
m_KIO_Safety_Ben = KIO_Fat_Ben + KIO_Inj_Ben + KIO_Prop_Ben

```

'Set module level values (to be displayed)

```

m_Values.Add m_KIO_TT_Min, "KIO_TT_Min"
m_Values.Add m_KIO_FrTT_Min, "KIO_FrTT_Min"
m_Values.Add m_KIO_TT_Ben, "KIO_TT_Ben"
m_Values.Add m_KIO_FrTT_Ben, "KIO_FrTT_Ben"

```

```

m_Values.Add m_KIO_CO_Tons, "KIO_CO_Tons"
m_Values.Add m_KIO_NOX_Tons, "KIO_NOX_Tons"
m_Values.Add m_KIO_PM10_Tons, "KIO_PM10_Tons"
m_Values.Add m_KIO_VOC_Tons, "KIO_Voc_Tons"
m_Values.Add m_KIO_Env_Ben, "KIO_Env_Ben"

```

```

m_Values.Add m_KIO_User_Ben, "KIO_User_Ben"
m_Values.Add m_KIO_User_Transfer, "KIO_User_Transfer"

```

```

m_Values.Add m_KIO_Fatality, "KIO_Fatality"
m_Values.Add m_KIO_Injury, "KIO_Injury"
m_Values.Add m_KIO_Property, "KIO_Property"
m_Values.Add m_KIO_Safety_Ben, "KIO_Safety_Ben"

```

'Calls local function to set cost nodes

```

CalculateCosts "KIO", Discount_Rate, N

```

```
End If
End If
```

```
If bCVOKio Then
```

```
Call SetSCRITSType("its_calc_SCRITS_CVOKio")
If Not m_rst.EOF Then
```

```
'-----
'CVO Traffic Information Kiosks
```

```
'Load Variable Message Sign Variables
```

```
CVOKio_No = m_rst("CVOKio_No")
CVOKio_TripPer_I = m_rst("CVOKio_TripPer_I")
CVOKio_TripPer_F = m_rst("CVOKio_TripPer_F")
CVOKio_Look = m_rst("CVOKio_Look")
CVOKio_SavePer = m_rst("CVOKio_SavPer")
CVOKio_SaveMin = m_rst("CVOKio_SavMin")
CVOKio_SaveMin = m_rst("CVOKio_AvgMin")
```

```
'Traffic and Travel
```

```
CVOKio_TripTot_I = CVOKio_No * CVOKio_TripPer_I
CVOKio_TripTot_F = CVOKio_No * CVOKio_TripPer_F
CVOKio_TripHr_I = CVOKio_TripTot_I * Avg_TripMin_BCI / 60
CVOKio_TripHr_F = CVOKio_TripTot_F * Avg_TripMin_BCF / 60
CVOKio_DySav_I = CVOKio_TripTot_I * CVOKio_Look * CVOKio_SavePer * CVOKio_SaveMin / 60
CVOKio_DySav_F = CVOKio_TripTot_F * CVOKio_Look * CVOKio_SavePer * CVOKio_SaveMin / 60
CVOKio_PerSav_I = CVOKio_DySav_I / CVOKio_TripHr_I
CVOKio_PerSav_F = CVOKio_DySav_F / CVOKio_TripHr_F
CVOKio_YrSav_I = KIO_DySav_I * Ann_Daily_Benefit
CVOKio_YrSav_F = KIO_DySav_F * Ann_Daily_Benefit
```

```
'Travel Time Calculations:
```

```
'Yearly Travel Time Benefits in Minutes (considers only freight users)
```

```
CVOKIO_TT_Min_I = CVOKio_YrSav_I * avo * 60
CVOKIO_TT_Min_F = CVOKio_YrSav_F * avo * 60
m_CVOKIO_TT_Min = (CVOKIO_TT_Min_F - CVOKIO_TT_Min_I) * N / 2
m_CVOKIO_FrTT_Min = CVOKio_TT_Min
```

```
'Travel Time Benefits in Dollars
```

```
CVOKio_FrTTBen_I = (CVOKIO_TT_Min_I) / 60 * Time_Value_Freight
CVOKio_FrTTBen_F = (CVOKIO_TT_Min_F) / 60 * Time_Value_Freight
CVOKio_FrTTBen_Grad = (CVOKIO_TT_Min_F - CVOKIO_TT_Min_I) / N
m_CVOKIO_TT_Ben = PofG * CVOKio_FrTTBen_Grad
m_CVOKIO_FrTT_Ben = CVOKio_TT_Ben
```

```
'Operating Cost Calculations
```

```
'SCRITS assumes no VMT change for CVOKio projects
m_CVOKIO_User_Ben = 0
m_CVOKIO_User_Transfer = 0
```

```
'Energy and Emissions
```

```
' (SCRITS assumes energy and emissions changes negligible for CVOKio)
m_CVOKIO_CO_Tons = 0
m_CVOKIO_NOX_Tons = 0
m_CVOKIO_PM10_Tons = 0
m_CVOKIO_VOC_Tons = 0
m_CVOKIO_Env_Ben = 0
```

```
'SAFETY Calculations
```

```
CVOKio_SecFatDy_I = FreeVMT_BCI * SecAccPer * Fat_Rate_Truck / 100000000
CVOKio_SecFatDy_F = FreeVMT_BCF * SecAccPer * Fat_Rate_Truck / 100000000
```

```

CVOKio_FatRedDy_I = CVOKio_PerSav_I * CVOKio_SecFatDy_I
CVOKio_FatRedDy_F = CVOKio_PerSav_F * CVOKio_SecFatDy_F
CVOKIO_Fatality_I = CVOKio_FatRedDy_I * Ann_Daily_Benefit
CVOKIO_Fatality_F = CVOKio_FatRedDy_F * Ann_Daily_Benefit
CVOKio_SecInjDy_I = FreeVMT_BCI * SecAccPer * Inj_Rate_Truck / 100000000
CVOKio_SecInjDy_F = FreeVMT_BCF * SecAccPer * Inj_Rate_Truck / 100000000
CVOKio_InjRedDy_I = CVOKio_PerSav_I * CVOKio_SecInjDy_I
CVOKio_InjRedDy_F = CVOKio_PerSav_F * CVOKio_SecInjDy_F
CVOKIO_Injury_I = CVOKio_InjRedDy_I * Ann_Daily_Benefit
CVOKIO_Injury_F = CVOKio_InjRedDy_F * Ann_Daily_Benefit
CVOKio_SecPropDy_I = FreeVMT_BCI * SecAccPer * Prop_Rate_Truck / 100000000
CVOKio_SecPropDy_F = FreeVMT_BCF * SecAccPer * Prop_Rate_Truck / 100000000
CVOKio_PropRedDy_I = CVOKio_PerSav_I * CVOKio_SecPropDy_I
CVOKio_PropRedDy_F = CVOKio_PerSav_F * CVOKio_SecPropDy_F
CVOKIO_Property_I = CVOKio_PropRedDy_I * Ann_Daily_Benefit
CVOKIO_Property_F = CVOKio_PropRedDy_F * Ann_Daily_Benefit

```

```

m_CVOKIO_Fatality = (CVOKIO_Fatality_F - CVOKIO_Fatality_I) * N / 2
m_CVOKIO_Injury = (CVOKIO_Injury_F - CVOKIO_Injury_I) * N / 2
m_CVOKIO_Property = (CVOKIO_Property_F - CVOKIO_Property_I) * N / 2

```

'Safety Benefit Calculations

```

CVOKIO_FatBen_Grad = (CVOKIO_Fatality_F - CVOKIO_Fatality_I) / N
CVOKIO_Fat_Ben = PofG * Fatality_Cost * CVOKIO_FatBen_Grad
CVOKIO_InjBen_Grad = (CVOKIO_Injury_F - CVOKIO_Injury_I) / N
CVOKIO_Inj_Ben = PofG * Evident_Cost * CVOKIO_InjBen_Grad
CVOKIO_PropBen_Grad = (CVOKIO_Property_F - CVOKIO_Property_I) / N
CVOKIO_Prop_Ben = PofG * PDO_Cost * CVOKIO_PropBen_Grad
m_CVOKIO_Safety_Ben = CVOKIO_Fat_Ben + CVOKIO_Inj_Ben + CVOKIO_Prop_Ben

```

'Set module level values (to be displayed)

```

m_Values.Add m_CVOKIO_TT_Min, "CVOKio_TT_Min"
m_Values.Add m_CVOKIO_FrTT_Min, "CVOKio_FrTT_Min"
m_Values.Add m_CVOKIO_TT_Ben, "CVOKio_TT_Ben"
m_Values.Add m_CVOKIO_FrTT_Ben, "CVOKio_FrTT_Ben"

m_Values.Add m_CVOKIO_CO_Tons, "CVOKio_CO_Tons"
m_Values.Add m_CVOKIO_NOX_Tons, "CVOKio_NOX_Tons"
m_Values.Add m_CVOKIO_PM10_Tons, "CVOKio_PM10_Tons"
m_Values.Add m_CVOKIO_VOC_Tons, "CVOKio_Voc_Tons"
m_Values.Add m_CVOKIO_Env_Ben, "CVOKio_Env_Ben"

m_Values.Add m_CVOKIO_User_Ben, "CVOKio_User_Ben"
m_Values.Add m_CVOKIO_User_Transfer, "CVOKio_User_Transfer"

m_Values.Add m_CVOKIO_Fatality, "CVOKio_Fatality"
m_Values.Add m_CVOKIO_Injury, "CVOKio_Injury"
m_Values.Add m_CVOKIO_Property, "CVOKio_Property"
m_Values.Add m_CVOKIO_Safety_Ben, "CVOKio_Safety_Ben"

```

'Calls local function to set cost nodes
CalculateCosts "CVOKio", Discount_Rate, N

End If
End If

If blnt Then
Call SetSCRITSType("its_calc_SCRITS_Int")
If Not m_rst.EOF Then

'-----

'Internet Information

'Load Variable Message Sign Variables

Int_TripPer_I = m_rst("INT_Per")
 Int_Look = m_rst("INT_Look")
 Int_SavePer = m_rst("INT_SavPer")
 Int_SaveMin = m_rst("INT_SavMin")

'Traffic and Travel

Avg_TripHr_I = AvgTripNo_BCI * Avg_TripMin_BCI / 60
 Avg_TripHr_F = AvgTripNo_BCF * Avg_TripMin_BCF / 60
 Int_TripTot_I = AvgTripNo_BCI * Int_TripPer_I
 Int_TripTot_F = AvgTripNo_BCF * Int_TripPer_I
 INT_DySav_I = Int_TripTot_I * Int_Look * Int_SavePer * Int_SaveMin / 60
 INT_DySav_F = Int_TripTot_F * Int_Look * Int_SavePer * Int_SaveMin / 60
 INT_PerSav_I = INT_DySav_I / Avg_TripHr_I
 INT_PerSav_F = INT_DySav_F / Avg_TripHr_F
 INT_YrSav_I = INT_DySav_I * Ann_Daily_Benefit
 INT_YrSav_F = INT_DySav_F * Ann_Daily_Benefit

'Travel Time Calculations:

'Yearly Travel Time Benefits in Minutes (freight considered in separate freight Intsk category)

'Assumes no induced travel

INT_TT_Min_I = INT_YrSav_I * avo * 60
 INT_TT_Min_F = INT_YrSav_F * avo * 60
 m_INT_TT_Min = (INT_TT_Min_F - INT_TT_Min_I) * N / 2
 m_INT_FrTT_Min = 0

'Travel Time Benefits in Dollars

INT_TT_Ben_I = (INT_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
 INT_TT_Ben_F = (INT_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
 INT_TTBen_Grad = (INT_TT_Ben_F - INT_TT_Ben_I) / N
 m_INT_TT_Ben = PofG * INT_TTBen_Grad
 m_INT_FrTT_Ben = 0

'Operating Cost Calculations

'SCRITS assumes no VMT change for INT projects

m_INT_User_Ben = 0
 m_INT_User_Transfer = 0

'Energy and Emissions

' (SCRITS assumes energy and emissions changes negligible for INT)

m_INT_CO_Tons = 0
 m_INT_NOX_Tons = 0
 m_INT_PM10_Tons = 0
 m_INT_VOC_Tons = 0
 m_INT_Env_Ben = 0

'SAFETY Calculations

INT_SecFatDy_I = FreeVMT_BCI * SecAccPer * Fat_Rate_Auto / 10000000
 INT_SecFatDy_F = FreeVMT_BCF * SecAccPer * Fat_Rate_Auto / 10000000
 INT_FatRedDy_I = INT_PerSav_I * INT_SecFatDy_I
 INT_FatRedDy_F = INT_PerSav_F * INT_SecFatDy_F
 INT_Fatality_I = INT_FatRedDy_I * Ann_Daily_Benefit
 INT_Fatality_F = INT_FatRedDy_F * Ann_Daily_Benefit
 INT_SecInjDy_I = FreeVMT_BCI * SecAccPer * Inj_Rate_Auto / 10000000
 INT_SecInjDy_F = FreeVMT_BCF * SecAccPer * Inj_Rate_Auto / 10000000
 INT_InjRedDy_I = INT_PerSav_I * INT_SecInjDy_I
 INT_InjRedDy_F = INT_PerSav_F * INT_SecInjDy_F
 INT_Injury_I = INT_InjRedDy_I * Ann_Daily_Benefit
 INT_Injury_F = INT_InjRedDy_F * Ann_Daily_Benefit
 INT_SecPropDy_I = FreeVMT_BCI * SecAccPer * Prop_Rate_Auto / 10000000

```

INT_SecPropDy_F = FreeVMT_BCF * SecAccPer * Prop_Rate_Auto / 10000000
INT_PropRedDy_I = INT_PerSav_I * INT_SecPropDy_I
INT_PropRedDy_F = INT_PerSav_F * INT_SecPropDy_F
INT_Property_I = INT_PropRedDy_I * Ann_Daily_Benefit
INT_Property_F = INT_PropRedDy_F * Ann_Daily_Benefit

```

```

m_INT_Fatality = (INT_Fatality_F - INT_Fatality_I) * N / 2
m_INT_Injury = (INT_Injury_F - INT_Injury_I) * N / 2
m_INT_Property = (INT_Property_F - INT_Property_I) * N / 2

```

'Safety Benefit Calculations

```

INT_FatBen_Grad = (INT_Fatality_F - INT_Fatality_I) / N
INT_Fat_Ben = PofG * Fatality_Cost * INT_FatBen_Grad
INT_InjBen_Grad = (INT_Injury_F - INT_Injury_I) / N
INT_Inj_Ben = PofG * Evident_Cost * INT_InjBen_Grad
INT_PropBen_Grad = (INT_Property_F - INT_Property_I) / N
INT_Prop_Ben = PofG * PDO_Cost * INT_PropBen_Grad
m_INT_Safety_Ben = INT_Fat_Ben + INT_Inj_Ben + INT_Prop_Ben

```

'Set module level values (to be displayed)

```

m_Values.Add m_INT_TT_Min, "INT_TT_Min"
m_Values.Add m_INT_FrTT_Min, "INT_FrTT_Min"
m_Values.Add m_INT_TT_Ben, "INT_TT_Ben"
m_Values.Add m_INT_FrTT_Ben, "INT_FrTT_Ben"

```

```

m_Values.Add m_INT_CO_Tons, "INT_CO_Tons"
m_Values.Add m_INT_NOX_Tons, "INT_NOX_Tons"
m_Values.Add m_INT_PM10_Tons, "INT_PM10_Tons"
m_Values.Add m_INT_VOC_Tons, "INT_Voc_Tons"
m_Values.Add m_INT_Env_Ben, "INT_Env_Ben"

```

```

m_Values.Add m_INT_User_Ben, "INT_User_Ben"
m_Values.Add m_INT_User_Transfer, "INT_User_Transfer"

```

```

m_Values.Add m_INT_Fatality, "INT_Fatality"
m_Values.Add m_INT_Injury, "INT_Injury"
m_Values.Add m_INT_Property, "INT_Property"
m_Values.Add m_INT_Safety_Ben, "INT_Safety_Ben"

```

'Calls local function to set cost nodes

```
CalculateCosts "INT", Discount_Rate, N
```

End If

End If

If bAVL Then

```
Call SetSCRITSType("its_calc_SCRITS_AVL")
```

```
If Not m_rst.EOF Then
```

```
'-----
```

'Automatic Vehicle Location (AVL) and Information

'Load Vehicle Location (AVL) Variables

```

AVL_WaitWO = m_rst("AVL_WaitWO")
AVL_WaitW = m_rst("AVL_WaitW")
AVL_WDBoard_I = m_rst("AVL_WDBoard_I")
AVL_WDBoard_F = m_rst("AVL_WDBoard_F")
AVL_FWBoard_I = m_rst("AVL_FWBoard_I")
AVL_FWBoard_F = m_rst("AVL_FWBoard_F")
AVL_Use_I = m_rst("AVL_Use_I")
AVL_Use_F = m_rst("AVL_Use_F")

```

```

'Traffic and Travel
AVL_WaitDif = (AVL_WaitWO - AVL_WaitW) / 60
AVL_DySavWD_I = AVL_WaitDif * AVL_WDBoard_I * AVL_Use_I
AVL_DySavWD_F = AVL_WaitDif * AVL_WDBoard_F * AVL_Use_F
AVL_DySavFW_I = AVL_WaitDif * AVL_FWBoard_I * AVL_Use_I
AVL_DySavFW_F = AVL_WaitDif * AVL_FWBoard_F * AVL_Use_F
AVL_YrSavWD_I = AVL_DySavWD_I * Ann_Daily_Benefit
AVL_YrSavWD_F = AVL_DySavWD_F * Ann_Daily_Benefit
AVL_YrSav_I = AVL_DySavFW_I * 365
AVL_YrSav_F = AVL_DySavFW_F * 365

'Travel Time Calculations:
'Yearly Travel Time Benefits in Minutes
'Assumes no induced travel
AVL_TT_Min_I = AVL_YrSav_I * 60
AVL_TT_Min_F = AVL_YrSav_F * 60
m_AVL_TT_Min = (AVL_TT_Min_F - AVL_TT_Min_I) * N / 2
m_AVL_FrTT_Min = 0

'Travel Time Benefits in Dollars
AVL_TT_Ben_I = (AVL_TT_Min_I) / 60 * (Percent_Time_Veh * Time_Value_Veh)
AVL_TT_Ben_F = (AVL_TT_Min_F) / 60 * (Percent_Time_Veh * Time_Value_Veh)
AVL_TTBen_Grad = (AVL_TT_Ben_F - AVL_TT_Ben_I) / N
m_AVL_TT_Ben = PofG * AVL_TTBen_Grad
m_AVL_FrTT_Ben = 0

'Operating Cost Calculations
'SCRITS assumes no VMT change for AVL projects
m_AVL_User_Ben = 0
m_AVL_User_Transfer = 0

'Energy and Emissions
'(SCRITS assumes energy and emissions changes negligible for AVL)
m_AVL_CO_Tons = 0
m_AVL_NOX_Tons = 0
m_AVL_PM10_Tons = 0
m_AVL_VOC_Tons = 0
m_AVL_Env_Ben = 0

'SAFETY Calculations
'Assumes safety benefits are negligible

m_AVL_Fatality = 0
m_AVL_Injury = 0
m_AVL_Property = 0

'Safety Benefit Calculations

m_AVL_Safety_Ben = 0

'Set module level values (to be displayed)
  m_Values.Add m_AVL_TT_Min, "AVL_TT_Min"
  m_Values.Add m_AVL_FrTT_Min, "AVL_FrTT_Min"
  m_Values.Add m_AVL_TT_Ben, "AVL_TT_Ben"
  m_Values.Add m_AVL_FrTT_Ben, "AVL_FrTT_Ben"

  m_Values.Add m_AVL_CO_Tons, "AVL_CO_Tons"
  m_Values.Add m_AVL_NOX_Tons, "AVL_NOX_Tons"
  m_Values.Add m_AVL_PM10_Tons, "AVL_PM10_Tons"
  m_Values.Add m_AVL_VOC_Tons, "AVL_Voc_Tons"
  m_Values.Add m_AVL_Env_Ben, "AVL_Env_Ben"

```

```

m_Values.Add m_AVL_User_Ben, "AVL_User_Ben"
m_Values.Add m_AVL_User_Transfer, "AVL_User_Transfer"

m_Values.Add m_AVL_Fatality, "Am_VL_Fatality"
m_Values.Add m_AVL_Injury, "AVL_Injury"
m_Values.Add m_AVL_Property, "AVL_Property"
m_Values.Add m_AVL_Safety_Ben, "AVL_Safety_Ben"

'Calls local function to set cost nodes
CalculateCosts "AVL", Discount_Rate, N

End If
End If

If bEFC Then
Call SetSCRITSType("its_calc_SCRITS_EFC")
If Not m_rst.EOF Then

'-----
'Bus Electronic Fare Collection

'Load Vehicle Location (AVL) Variables
EFC_BusSpd = m_rst("EFC_BusSpd")
EFC_BrdPer = m_rst("EFC_BrdPer")
EFC_MinCF = m_rst("EFC_MinCF")
EFC_MinEF = m_rst("EFC_MinEF")
EFC_PerEFWO = m_rst("EFC_PerEFWO")
EFC_PerEFW_I = m_rst("EFC_PerEFW_I")
EFC_PerEFW_F = m_rst("EFC_PerEFW_F")
EFC_WDPass_I = m_rst("EFC_WDPass_I")
EFC_WDPass_F = m_rst("EFC_WDPass_F")
EFC_FWPass_I = m_rst("EFC_FWPass_I")
EFC_FWPass_F = m_rst("EFC_FWPass_F")
EFC_TripLen = m_rst("EFC_TripLen")
EFC_Elas = m_rst("EFC_Elas")

'Traffic and Travel
EFC_SpdMin = 1 / EFC_BusSpd * 60
EFC_MinDif = EFC_MinCF - EFC_MinEF
EFC_SpdMinW_I = EFC_SpdMin - (EFC_SpdMin * EFC_BrdPer * EFC_MinDif / EFC_MinCF *
EFC_PerEFW_I)
EFC_SpdMinW_F = EFC_SpdMin - (EFC_SpdMin * EFC_BrdPer * EFC_MinDif / EFC_MinCF *
EFC_PerEFW_F)
EFC_SpdW_I = 1 / EFC_SpdMinW_I * 60
EFC_SpdW_F = 1 / EFC_SpdMinW_F * 60
EFC_SpdInc_I = (EFC_SpdW_I - EFC_BusSpd) / EFC_BusSpd
EFC_SpdInc_F = (EFC_SpdW_F - EFC_BusSpd) / EFC_BusSpd
EFC_WDHrsWO_I = EFC_WDPass_I * EFC_TripLen / EFC_BusSpd
EFC_WDHrsWO_F = EFC_WDPass_F * EFC_TripLen / EFC_BusSpd
EFC_WDHrsW_I = EFC_WDPass_I * EFC_TripLen / EFC_SpdW_I
EFC_WDHrsW_F = EFC_WDPass_F * EFC_TripLen / EFC_SpdW_F
EFC_DySavWD_I = EFC_WDHrsWO_I - EFC_WDHrsW_I
EFC_DySavWD_F = EFC_WDHrsWO_F - EFC_WDHrsW_F
'EFC_DySavFW_I = EFC_FWHrsWO_I - EFC_FWHrsW_I
'EFC_DySavFW_F = EFC_FWHrsWO_F - EFC_FWHrsW_F
EFC_YrSavWD_I = EFC_DySavWD_I * Ann_Daily_Benefit
EFC_YrSavWD_F = EFC_DySavWD_F * Ann_Daily_Benefit
'EFC_YrSav_I = EFC_DySavFW_I * 365
'EFC_YrSav_F = EFC_DySavFW_F * 365

'Induced Bus Travel

```

```

EFC_WDPassInc_I = EFC_WDPass_I * EFC_SpdInc_I * EFC_Elas
EFC_WDPassInc_F = EFC_WDPass_F * EFC_SpdInc_F * EFC_Elas
EFC_FWPassInc_I = EFC_FWPass_I * EFC_SpdInc_I * EFC_Elas
EFC_FWPassInc_F = EFC_FWPass_F * EFC_SpdInc_F * EFC_Elas
EFC_PerRed_I = EFC_WDPassInc_I / AvgTripNo_BCI
EFC_PerRed_F = EFC_WDPassInc_F / AvgTripNo_BCF
EFC_VMTSav_I = EFC_FWPassInc_I * Avg_TripLen_BCI
EFC_VMTSav_F = EFC_FWPassInc_F * Avg_TripLen_BCF

```

'Travel Time Calculations:

'Yearly Travel Time Benefits in Minutes

```

EFC_TT_Min_I = EFC_YrSavWD_I * 60
EFC_TT_Min_F = EFC_YrSavWD_F * 60
m_EFC_TT_Min = (EFC_TT_Min_F - EFC_TT_Min_I) * N / 2
m_EFC_FrTT_Min = 0

```

'Travel Time Benefits in Dollars

```

EFC_TT_Ben_I = (EFC_TT_Min_I) / 60 * (Percent_Time_Veh * Time_Value_Veh)
EFC_TT_Ben_F = (EFC_TT_Min_F) / 60 * (Percent_Time_Veh * Time_Value_Veh)
EFC_TTBen_Grad = (EFC_TT_Ben_F - EFC_TT_Ben_I) / N
m_EFC_TT_Ben = PofG * EFC_TTBen_Grad
m_EFC_FrTT_Ben = 0

```

'Operating Cost Calculations

```

EFC_VMT_I = EFC_VMTSav_I * 365
EFC_VMT_F = EFC_VMTSav_F * 365
EFC_VMT_Tot = (EFC_VMT_F - EFC_VMT_I) * N / 2

```

'User Cost Benefit Calculations

If Full_Cost Then

```

    EFC_UserBen_I = (EFC_VMT_I * Veh_OpCost_Full)
    EFC_UserBen_F = (EFC_VMT_F * Veh_OpCost_Full)

```

Else

```

    EFC_UserBen_I = (EFC_VMT_I * Veh_OpCost_Direct)
    EFC_UserBen_F = (EFC_VMT_F * Veh_OpCost_Direct)

```

End If

```

EFC_UCBen_Grad = (EFC_UserBen_F - EFC_UserBen_I) / N
m_EFC_User_Ben = PofG * EFC_UCBen_Grad
EFC_RevInc_I = EFC_FWPassInc_I * 365 * Avg_BusFare
EFC_RevInc_F = EFC_FWPassInc_F * 365 * Avg_BusFare
EFC_RevInc_Grad = (EFC_RevInc_F - EFC_RevInc_I) / N
m_EFC_User_Transfer = PofG * EFC_RevInc_Grad

```

'Energy and Emissions

' (Does not currently differentiate between autos and freight vehicles.)

'CO Calculations

```

EFC_CORate_BCI = m_Ikup_Pollution.RetrieveRate(ArtSpeed_BCI, em_CO, veht_auto)
EFC_CORate_BCF = m_Ikup_Pollution.RetrieveRate(ArtSpeed_BCF, em_CO, veht_auto)
EFC_CO_Gram_BCI = (EFC_VMTSav_I * 365 * EFC_CORate_BCI)
EFC_CO_Gram_BCF = (EFC_VMTSav_F * 365 * EFC_CORate_BCF)
EFC_CO_Gram = (EFC_CO_Gram_BCI + EFC_CO_Gram_BCF) / 2 * N
m_EFC_CO_Tons = EFC_CO_Gram / 907184.74

```

'NOX Calculations

```

EFC_NOXRate_BCI = m_Ikup_Pollution.RetrieveRate(ArtSpeed_BCI, em_NOX, veht_auto)
EFC_NOXRate_BCF = m_Ikup_Pollution.RetrieveRate(ArtSpeed_BCF, em_NOX, veht_auto)
EFC_NOX_Gram_BCI = (EFC_VMTSav_I * 365 * EFC_NOXRate_BCI)
EFC_NOX_Gram_BCF = (EFC_VMTSav_F * 365 * EFC_NOXRate_BCF)
EFC_NOX_Gram = (EFC_NOX_Gram_BCI + EFC_NOX_Gram_BCF) / 2 * N

```

```
m_EFC_NOX_Tons = EFC_NOX_Gram / 907184.74
```

```
'PM10 Calculations
```

```
EFC_PM10_Gram_BCI = (EFC_VMTSav_I * 365 * PM10_Rate_Auto)
```

```
EFC_PM10_Gram_BCF = (EFC_VMTSav_F * 365 * PM10_Rate_Auto)
```

```
EFC_PM10_Gram = (EFC_PM10_Gram_BCI + EFC_PM10_Gram_BCF) / 2 * N
```

```
m_EFC_PM10_Tons = EFC_PM10_Gram / 907184.74
```

```
'VOC Calculations
```

```
EFC_VOC_Gram_BCI = (EFC_VMTSav_I * 365 * VOC_Rate_Auto)
```

```
EFC_VOC_Gram_BCF = (EFC_VMTSav_F * 365 * VOC_Rate_Auto)
```

```
EFC_VOC_Gram = (EFC_VOC_Gram_BCI + EFC_VOC_Gram_BCF) / 2 * N
```

```
m_EFC_VOC_Tons = EFC_VOC_Gram / 907184.74
```

```
'Emissions Benefit Calculations
```

```
EFC_CO_Grad = (EFC_CO_Gram_BCF - EFC_CO_Gram_BCI) / N
```

```
EFC_CO_Ben = EFC_CO_Grad * PofG * COTon_Cost / 907184.74
```

```
EFC_NOX_Grad = (EFC_NOX_Gram_BCF - EFC_NOX_Gram_BCI) / N
```

```
EFC_NOX_Ben = EFC_NOX_Grad * PofG * NOXTon_Cost / 907184.74
```

```
EFC_PM10_Grad = (EFC_PM10_Gram_BCF - EFC_PM10_Gram_BCI) / N
```

```
EFC_PM10_Ben = EFC_PM10_Grad * PofG * PM10Ton_Cost / 907184.74
```

```
EFC_VOC_Grad = (EFC_VOC_Gram_BCF - EFC_VOC_Gram_BCI) / N
```

```
EFC_VOC_Ben = EFC_VOC_Grad * PofG * VOCTon_Cost / 907184.74
```

```
m_EFC_Env_Ben = EFC_CO_Ben + EFC_NOX_Ben + EFC_PM10_Ben + EFC_VOC_Ben
```

```
'SAFETY Calculations
```

```
'Assumes safety benefits are negligible
```

```
m_EFC_Fatality = 0
```

```
m_EFC_Injury = 0
```

```
m_EFC_Property = 0
```

```
'Safety Benefit Calculations
```

```
m_EFC_Safety_Ben = 0
```

```
'Set module level values (to be displayed)
```

```
  m_Values.Add m_EFC_TT_Min, "EFC_TT_Min"
```

```
  m_Values.Add m_EFC_FrTT_Min, "EFC_FrTT_Min"
```

```
  m_Values.Add m_EFC_TT_Ben, "EFC_TT_Ben"
```

```
  m_Values.Add m_EFC_FrTT_Ben, "EFC_FrTT_Ben"
```

```
  m_Values.Add m_EFC_CO_Tons, "EFC_CO_Tons"
```

```
  m_Values.Add m_EFC_NOX_Tons, "EFC_NOX_Tons"
```

```
  m_Values.Add m_EFC_PM10_Tons, "EFC_PM10_Tons"
```

```
  m_Values.Add m_EFC_VOC_Tons, "EFC_Voc_Tons"
```

```
  m_Values.Add m_EFC_Env_Ben, "EFC_Env_Ben"
```

```
  m_Values.Add m_EFC_User_Ben, "EFC_User_Ben"
```

```
  m_Values.Add m_EFC_User_Transfer, "EFC_User_Transfer"
```

```
  m_Values.Add m_EFC_Fatality, "EFC_Fatality"
```

```
  m_Values.Add m_EFC_Injury, "EFC_Injury"
```

```
  m_Values.Add m_EFC_Property, "EFC_Property"
```

```
  m_Values.Add m_EFC_Safety_Ben, "EFC_Safety_Ben"
```

```
'Calls local function to set cost nodes
```

```

    CalculateCosts "EFC", Discount_Rate, N

    End If
End If

If bBPS Then
    Call SetSCRITSType("its_calc_SCRITS_BPS")
    If Not m_rst.EOF Then

        '-----
        'Bus Priority Systems (BPS)

        'Load Bus Priority Systems Variables
        BPS_Miles = m_rst("BPS_Miles")
        BPS_BusNo = m_rst("BPS_BusNo")
        BPS_BusSpd = m_rst("BPS_BusSpd")
        BPS_PerSig = m_rst("BPS_PerSig")
        BPS_PerRed = m_rst("BPS_PerRed")
        BPS_Pass_I = m_rst("BPS_Pass_I")
        BPS_Pass_F = m_rst("BPS_Pass_F")
        BPS_TripLen = m_rst("BPS_TripLen")
        BPS_Elas = m_rst("BPS_Elas")
        BPS_Veh_I = m_rst("BPS_Veh_I")
        BPS_Veh_F = m_rst("BPS_Veh_F")
        BPS_Cross_I = m_rst("BPS_Cross_I")
        BPS_Cross_F = m_rst("BPS_Cross_F")
        BPS_PerDelay = m_rst("BPS_PerDelay")
        BPS_MinDelay = m_rst("BPS_MinDelay")

        'Traffic and Travel
        BPS_SpdMin = 1 / BPS_BusSpd * 60
        BPS_SpdMinW = BPS_SpdMin - (BPS_SpdMin * BPS_PerSig * BPS_PerRed)
        BPS_SpdW = 1 / BPS_SpdMinW * 60
        BPS_SpdInc = (BPS_SpdW - BPS_BusSpd) / BPS_BusSpd
        BPS_RtHrsDy = BPS_Miles * BPS_BusNo / BPS_BusSpd - BPS_Miles * BPS_Ni / BPS_SpdW
        BPS_YrRtHrs_I = BPS_RtHrsDy * Ann_Daily_Benefit
        BPS_WDHrsWO_I = BPS_Pass_I * BPS_TripLen / BPS_BusSpd
        BPS_WDHrsWO_F = BPS_Pass_F * BPS_TripLen / BPS_BusSpd
        BPS_WDHrsW_I = BPS_Pass_I * BPS_TripLen / BPS_SpdW
        BPS_WDHrsW_F = BPS_Pass_F * BPS_TripLen / BPS_SpdW
        BPS_WDSav_I = BPS_WDHrsWO_I - BPS_WDHrsW_I
        BPS_WDSav_F = BPS_WDHrsWO_F - BPS_WDHrsW_F
        BPS_YrSav_I = BPS_WDSav_I * Ann_Daily_Benefit
        BPS_YrSav_F = BPS_WDSav_F * Ann_Daily_Benefit

        'Induced Bus Travel
        BPS_PassInc_I = BPS_Pass_I * BPS_SpdInc * BPS_Elas
        BPS_PassInc_F = BPS_Pass_F * BPS_SpdInc * BPS_Elas
        BPS_VehRedPer_I = BPS_PassInc_I / BPS_Veh_I
        BPS_VehRedPer_F = BPS_PassInc_F / BPS_Veh_F
        BPS_AddVehHrs_I = BPS_Cross_I * BPS_MinDelay / 3600
        BPS_AddVehHrs_F = BPS_Cross_F * BPS_MinDelay / 3600
        BPS_PerRed_I = BPS_PassInc_I / AvgTripNo_BCI
        BPS_PerRed_F = BPS_PassInc_F / AvgTripNo_BCF
        BPS_VMTSav_I = BPS_PassInc_I * Avg_TripLen_BCI
        BPS_VMTSav_F = BPS_PassInc_F * Avg_TripLen_BCF

        'Travel Time Calculations:
        'Yearly Travel Time Benefits in Minutes
        BPS_TT_Min_I = (BPS_YrSav_I * 60) - (BPS_AddVehHrs_I * 60 * avo)
        BPS_TT_Min_F = BPS_YrSav_F * 60 - (BPS_AddVehHrs_I * 60 * avo)

```

```

m_BPS_TT_Min = (BPS_TT_Min_F - BPS_TT_Min_I) * N / 2
m_BPS_FrTT_Min = 0

'Travel Time Benefits in Dollars
BPS_TT_Ben_I = (BPS_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
BPS_TT_Ben_F = (BPS_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
BPS_TTBen_Grad = (BPS_TT_Ben_F - BPS_TT_Ben_I) / N
m_BPS_TT_Ben = PofG * BPS_TTBen_Grad
m_BPS_FrTT_Ben = 0

'Operating Cost Calculations
BPS_VMT_I = BPS_VMTSav_I * Ann_Daily_Benefit
BPS_VMT_F = BPS_VMTSav_F * Ann_Daily_Benefit
BPS_VMT_Tot = (BPS_VMT_F - BPS_VMT_I) * N / 2

'User Cost Benefit Calculations (does not currently consider bus operating savings)
If Full_Cost Then
    BPS_UserBen_I = (BPS_VMT_I * Veh_OpCost_Full)
    BPS_UserBen_F = (BPS_VMT_F * Veh_OpCost_Full)
Else
    BPS_UserBen_I = (BPS_VMT_I * Veh_OpCost_Direct)
    BPS_UserBen_F = (BPS_VMT_F * Veh_OpCost_Direct)
End If

BPS_UCBen_Grad = (BPS_UserBen_F - BPS_UserBen_I) / N
m_BPS_User_Ben = PofG * BPS_UCBen_Grad
BPS_RevInc_I = BPS_PassInc_I * Ann_Daily_Benefit * Avg_BusFare
BPS_RevInc_F = BPS_PassInc_F * Ann_Daily_Benefit * Avg_BusFare
BPS_RevInc_Grad = (BPS_RevInc_F - BPS_RevInc_I) / N
m_BPS_User_Transfer = PofG * BPS_RevInc_Grad

'Energy and Emissions
'(Does not currently differentiate between autos and freight vehicles.)
'CO Calculations
BPS_CORate_BCI = m_Ikup_Pollution.RetrieveRate(ArtSpeed_BCI, em_CO, veht_auto)
BPS_CORate_BCF = m_Ikup_Pollution.RetrieveRate(ArtSpeed_BCF, em_CO, veht_auto)
BPS_CO_Gram_BCI = (BPS_VMTSav_I * 365 * BPS_CORate_BCI)
BPS_CO_Gram_BCF = (BPS_VMTSav_F * 365 * BPS_CORate_BCF)
BPS_CO_Gram = (BPS_CO_Gram_BCI + BPS_CO_Gram_BCF) / 2 * N
m_BPS_CO_Tons = BPS_CO_Gram / 907184.74

'NOX Calculations
BPS_NOXRate_BCI = m_Ikup_Pollution.RetrieveRate(ArtSpeed_BCI, em_NOX, veht_auto)
BPS_NOXRate_BCF = m_Ikup_Pollution.RetrieveRate(ArtSpeed_BCF, em_NOX, veht_auto)
BPS_NOX_Gram_BCI = (BPS_VMTSav_I * 365 * BPS_NOXRate_BCI)
BPS_NOX_Gram_BCF = (BPS_VMTSav_F * 365 * BPS_NOXRate_BCF)
BPS_NOX_Gram = (BPS_NOX_Gram_BCI + BPS_NOX_Gram_BCF) / 2 * N
m_BPS_NOX_Tons = BPS_NOX_Gram / 907184.74

'PM10 Calculations
BPS_PM10_Gram_BCI = (BPS_VMTSav_I * 365 * PM10_Rate_Auto)
BPS_PM10_Gram_BCF = (BPS_VMTSav_F * 365 * PM10_Rate_Auto)
BPS_PM10_Gram = (BPS_PM10_Gram_BCI + BPS_PM10_Gram_BCF) / 2 * N
m_BPS_PM10_Tons = BPS_PM10_Gram / 907184.74

'VOC Calculations
BPS_VOC_Gram_BCI = (BPS_VMTSav_I * 365 * VOC_Rate_Auto)
BPS_VOC_Gram_BCF = (BPS_VMTSav_F * 365 * VOC_Rate_Auto)
BPS_VOC_Gram = (BPS_VOC_Gram_BCI + BPS_VOC_Gram_BCF) / 2 * N
m_BPS_VOC_Tons = BPS_VOC_Gram / 907184.74
'Emissions Benefit Calculations
BPS_CO_Grad = (BPS_CO_Gram_BCF - BPS_CO_Gram_BCI) / N

```



```
BPS_CO_Ben = BPS_CO_Grad * PofG * CO_Ton_Cost / 907184.74
```

```
BPS_NOX_Grad = (BPS_NOX_Gram_BCF - BPS_NOX_Gram_BCI) / N
BPS_NOX_Ben = BPS_NOX_Grad * PofG * NOX_Ton_Cost / 907184.74
```

```
BPS_PM10_Grad = (BPS_PM10_Gram_BCF - BPS_PM10_Gram_BCI) / N
BPS_PM10_Ben = BPS_PM10_Grad * PofG * PM10_Ton_Cost / 907184.74
```

```
BPS_VOC_Grad = (BPS_VOC_Gram_BCF - BPS_VOC_Gram_BCI) / N
BPS_VOC_Ben = BPS_VOC_Grad * PofG * VOC_Ton_Cost / 907184.74
```

```
m_BPS_Env_Ben = BPS_CO_Ben + BPS_NOX_Ben + BPS_PM10_Ben + BPS_VOC_Ben
```

```
'SAFETY Calculations
```

```
'Assumes safety benefits are negligible
```

```
m_BPS_Fatality = 0
```

```
m_BPS_Injury = 0
```

```
m_BPS_Property = 0
```

```
'Safety Benefit Calculations
```

```
m_BPS_Safety_Ben = 0
```

```
'Set module level values (to be displayed)
```

```
  m_Values.Add m_BPS_TT_Min, "BPS_TT_Min"
```

```
  m_Values.Add m_BPS_FrTT_Min, "BPS_FrTT_Min"
```

```
  m_Values.Add m_BPS_TT_Ben, "BPS_TT_Ben"
```

```
  m_Values.Add m_BPS_FrTT_Ben, "BPS_FrTT_Ben"
```

```
  m_Values.Add m_BPS_CO_Tons, "BPS_CO_Tons"
```

```
  m_Values.Add m_BPS_NOX_Tons, "BPS_NOX_Tons"
```

```
  m_Values.Add m_BPS_PM10_Tons, "BPS_PM10_Tons"
```

```
  m_Values.Add m_BPS_VOC_Tons, "BPS_Voc_Tons"
```

```
  m_Values.Add m_BPS_Env_Ben, "BPS_Env_Ben"
```

```
  m_Values.Add m_BPS_User_Ben, "BPS_User_Ben"
```

```
  m_Values.Add m_BPS_User_Transfer, "BPS_User_Transfer"
```

```
  m_Values.Add m_BPS_Fatality, "BPS_Fatality"
```

```
  m_Values.Add m_BPS_Injury, "BPS_Injury"
```

```
  m_Values.Add m_BPS_Property, "BPS_Property"
```

```
  m_Values.Add m_BPS_Safety_Ben, "BPS_Safety_Ben"
```

```
'Calls local function to set cost nodes
```

```
CalculateCosts "BPS", Discount_Rate, N
```

```
End If
```

```
End If
```

```
If bETC Then
```

```
  Call SetSCRITSType("its_calc_SCRITS_ETC")
```

```
  If Not m_rst.EOF Then
```

```
    '-----
    'Electronic Toll Collection (ETC)
```

```
    'Load Electronic Toll Collection (ETC) Variables
```

```
    ETC_Vol_I = m_rst("ETC_Vol_I")
```

```
    ETC_Vol_F = m_rst("ETC_Vol_F")
```

```
    ETC_WOEx = m_rst("ETC_WOEx")
```

```

ETC_WOReg = m_rst("ETC_WOReg")
ETC_WOOTH = m_rst("ETC_WOOTH")
ETC_WET = m_rst("ETC_WET")
ETC_WEX = m_rst("ETC_WEX")
ETC_WReg = m_rst("ETC_WReg")
ETC_WOTH = m_rst("ETC_WOTH")
ETC_ETSec = m_rst("ETC_ETSec")
ETC_ExSec = m_rst("ETC_ExSec")
ETC_RegSec = m_rst("ETC_RegSec")
ETC_OthSec = m_rst("ETC_OthSec")

```

'Traffic and Travel

```

ETC_HrsWO_I = ETC_Vol_I * ((ETC_WOEx * ETC_ExSec) + (ETC_WOReg * ETC_RegSec) +
(ETC_WOOTH * ETC_OthSec)) / 3600
ETC_HrsWO_F = ETC_Vol_F * ((ETC_WOEx * ETC_ExSec) + (ETC_WOReg * ETC_RegSec) +
(ETC_WOOTH * ETC_OthSec)) / 3600
ETC_HrsW_I = ETC_Vol_I * ((ETC_WET * ETC_ETSec) + (ETC_WEX * ETC_ExSec) + (ETC_WReg *
ETC_RegSec) + (ETC_WOTH * ETC_OthSec)) / 3600
ETC_HrsW_F = ETC_Vol_F * ((ETC_WET * ETC_ETSec) + (ETC_WEX * ETC_ExSec) + (ETC_WReg
* ETC_RegSec) + (ETC_WOTH * ETC_OthSec)) / 3600
ETC_WDSav_I = ETC_HrsWO_I - ETC_HrsW_I
ETC_WDSav_F = ETC_HrsWO_F - ETC_HrsW_F
ETC_YrSav_I = ETC_WDSav_I * Ann_Daily_Benefit
ETC_YrSav_F = ETC_WDSav_F * Ann_Daily_Benefit

```

'Travel Time Calculations:

'Yearly Travel Time Benefits in Minutes

```

ETC_TT_Min_I = (ETC_YrSav_I * 60)
ETC_TT_Min_F = ETC_YrSav_F * 60
m_ETC_TT_Min = (ETC_TT_Min_F - ETC_TT_Min_I) * N / 2
m_ETC_FrTT_Min = 0

```

'Travel Time Benefits in Dollars

```

ETC_TT_Ben_I = (ETC_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
ETC_TT_Ben_F = (ETC_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
ETC_TTBen_Grad = (ETC_TT_Ben_F - ETC_TT_Ben_I) / N
m_ETC_TT_Ben = PofG * ETC_TTBen_Grad
m_ETC_FrTT_Ben = 0

```

'Operating Cost Calculations

'(No change in VMT)

```

m_ETC_User_Ben = 0
m_ETC_User_Transfer = 0

```

'Energy and Emissions

'(No change in VMT)

```

m_ETC_CO_Tons = 0
m_ETC_NOX_Tons = 0
m_ETC_PM10_Tons = 0
m_ETC_VOC_Tons = 0
m_ETC_Env_Ben = 0

```

'SAFETY Calculations

'Assumes safety benefits are negligible

```

m_ETC_Fatality = 0
m_ETC_Injury = 0
m_ETC_Property = 0

```

'Safety Benefit Calculations

```

m_ETC_Safety_Ben = 0

'Set module level values (to be displayed)
  m_Values.Add m_ETC_TT_Min, "ETC_TT_Min"
  m_Values.Add m_ETC_FrTT_Min, "ETC_FrTT_Min"
  m_Values.Add m_ETC_TT_Ben, "ETC_TT_Ben"
  m_Values.Add m_ETC_FrTT_Ben, "ETC_FrTT_Ben"

  m_Values.Add m_ETC_CO_Tons, "ETC_CO_Tons"
  m_Values.Add m_ETC_NOX_Tons, "ETC_NOX_Tons"
  m_Values.Add m_ETC_PM10_Tons, "ETC_PM10_Tons"
  m_Values.Add m_ETC_VOC_Tons, "ETC_Voc_Tons"
  m_Values.Add m_ETC_Env_Ben, "ETC_Env_Ben"

  m_Values.Add m_ETC_User_Ben, "ETC_User_Ben"
  m_Values.Add m_ETC_User_Transfer, "ETC_User_Transfer"

  m_Values.Add m_ETC_Fatality, "ETC_Fatality"
  m_Values.Add m_ETC_Injury, "ETC_Injury"
  m_Values.Add m_ETC_Property, "ETC_Property"
  m_Values.Add m_ETC_Safety_Ben, "ETC_Safety_Ben"

'Calls local function to set cost nodes
CalculateCosts "ETC", Discount_Rate, N

End If
End If

If bRamp Then
  Call SetSCRITSType("its_calc_SCRITS_Ramp")
  If Not m_rst.EOF Then

    '-----
    'Ramp Metering

    'Load Ramp Metering Variables
    Ramp_AnNo = m_rst("Ramp_AnNo")
    Ramp_PerAccRed = m_rst("Ramp_PerAccRed")

    'Traffic and Travel

    'Begin Calculation Loop.

    While Not m_rst.EOF

      'Load Ramp Variables
      Ramp_PkPer = m_rst("Ramp_PkPer")
      Ramp_Dir = m_rst("Ramp_Dir")
      Ramp_PerStudy = m_rst("Ramp_PerStudy")
      Ramp_PerPkVol = m_rst("Ramp_PerPkVol")
      Ramp_PerPkDir = m_rst("Ramp_PerPkDir")
      Ramp_SpdWO = m_rst("Ramp_SpdWO")
      Ramp_SpdW = m_rst("Ramp_SpdW")
      Ramp_No = m_rst("Ramp_No")
      Ramp_RampVol = m_rst("Ramp_RampVol")
      Ramp_Delay = m_rst("Ramp_Delay")
      Ramp_ArtPkVol = m_rst("Ramp_ArtPkVol")
      Ramp_ArtPkDir = m_rst("Ramp_ArtPkDir")
      Ramp_ArtSpdWO = m_rst("Ramp_ArtSpdWO")
      Ramp_ArtSpdW = m_rst("Ramp_ArtSpdW")

      'Traffic and Travel - Freeway

```

```

Ramp_FreeVmt_I = FreeVMT_BCI * Ramp_PerStudy * Ramp_PerPkVol * Ramp_PerPkDir
Ramp_FreeVmt_F = FreeVMT_BCF * Ramp_PerStudy * Ramp_PerPkVol * Ramp_PerPkDir
Ramp_FhrsWO_I = Ramp_FreeVmt_I / Ramp_SpdWO
Ramp_FhrsWO_F = Ramp_FreeVmt_F / Ramp_SpdWO
Ramp_FhrsW_I = Ramp_FreeVmt_I / Ramp_SpdW
Ramp_FhrsW_F = Ramp_FreeVmt_F / Ramp_SpdW
Ramp_FDySav_I = Ramp_FhrsWO_I - Ramp_FhrsW_I
Ramp_FDySav_F = Ramp_FhrsWO_F - Ramp_FhrsW_F
Ramp_RampSav = Ramp_No * Ramp_RampVol * Ramp_Delay

'Traffic and Travel - Arterial
Ramp_ArtLen = Ramp_PerStudy * Free_miles
Ramp_ArtVmt_I = Ramp_PerPkDir * Ramp_ArtLen * Ramp_ArtPkVol * Ramp_ArtPkDir *
ArtVMT_BCI
Ramp_ArtVmt_F = Ramp_PerPkDir * Ramp_ArtLen * Ramp_ArtPkVol * Ramp_ArtPkDir *
ArtVMT_BCF
Ramp_AhrsWO_I = Ramp_ArtVmt_I / Ramp_ArtSpdWO
Ramp_AhrsWO_F = Ramp_ArtVmt_F / Ramp_ArtSpdWO
Ramp_AhrsW_I = Ramp_ArtVmt_I / Ramp_ArtSpdW
Ramp_AhrsW_F = Ramp_ArtVmt_F / Ramp_ArtSpdW
Ramp_AdySav_I = Ramp_AhrsWO_I - Ramp_AhrsW_I
Ramp_AdySav_F = Ramp_AhrsWO_F - Ramp_AhrsW_F
Ramp_DySav_I = Ramp_FDySav_I + Ramp_RampSav + Ramp_AdySav_I
Ramp_DySav_F = Ramp_FDySav_F + Ramp_RampSav + Ramp_AdySav_F
Ramp_YrSav_I = Ramp_DySav_I * Ann_Daily_Benefit
Ramp_YrSav_F = Ramp_DySav_F * Ann_Daily_Benefit

'Get Sum
Ramp_YrSav_I = Ramp_YrSav_I + Ramp_YrSav_I
Ramp_YrSav_F = Ramp_YrSav_F + Ramp_YrSav_F

m_rst.MoveNext
Wend

'Reset recordset
m_rst.MoveFirst

'Travel Time Calculations:
'Yearly Travel Time Benefits in Minutes
RAMP_TT_Min_I = (Ramp_YrSav_I * 60)
RAMP_TT_Min_F = Ramp_YrSav_F * 60
m_RAMP_TT_Min = (RAMP_TT_Min_F - RAMP_TT_Min_I) * N / 2
m_RAMP_FrTT_Min = 0

'Travel Time Benefits in Dollars
RAMP_TT_Ben_I = (RAMP_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
RAMP_TT_Ben_F = (RAMP_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
RAMP_TTBen_Grad = (RAMP_TT_Ben_F - RAMP_TT_Ben_I) / N
m_RAMP_TT_Ben = PofG * RAMP_TTBen_Grad
m_RAMP_FrTT_Ben = 0

'Operating Cost Calculations
'(No change in VMT)
m_RAMP_User_Ben = 0
m_RAMP_User_Transfer = 0

'Energy and Emissions
'(No change in VMT)
m_RAMP_CO_Tons = 0
m_RAMP_NOX_Tons = 0
m_RAMP_PM10_Tons = 0
m_RAMP_VOC_Tons = 0

```

```

m_RAMP_Env_Ben = 0

'SAFETY Calculations
'Assumes safety benefits are negligible

m_RAMP_Fatality = 0
m_RAMP_Injury = 0
m_RAMP_Property = 0

'Safety Benefit Calculations

m_RAMP_Safety_Ben = 0

'Set module level values (to be displayed)
  m_Values.Add m_RAMP_TT_Min, "Ramp_TT_Min"
  m_Values.Add m_RAMP_FrTT_Min, "Ramp_FrTT_Min"
  m_Values.Add m_RAMP_TT_Ben, "Ramp_TT_Ben"
  m_Values.Add m_RAMP_FrTT_Ben, "Ramp_FrTT_Ben"

  m_Values.Add m_RAMP_CO_Tons, "Ramp_CO_Tons"
  m_Values.Add m_RAMP_NOX_Tons, "Ramp_NOX_Tons"
  m_Values.Add m_RAMP_PM10_Tons, "Ramp_PM10_Tons"
  m_Values.Add m_RAMP_VOC_Tons, "Ramp_Voc_Tons"
  m_Values.Add m_RAMP_Env_Ben, "Ramp_Env_Ben"

  m_Values.Add m_RAMP_User_Ben, "Ramp_User_Ben"
  m_Values.Add m_RAMP_User_Transfer, "Ramp_User_Transfer"

  m_Values.Add m_RAMP_Fatality, "Ramp_Fatality"
  m_Values.Add m_RAMP_Injury, "Ramp_Injury"
  m_Values.Add m_RAMP_Property, "Ramp_Property"
  m_Values.Add m_RAMP_Safety_Ben, "Ramp_Safety_Ben"

'Calls local function to set cost nodes
CalculateCosts "Ramp", Discount_Rate, N

End If
End If

If bWIM Then
  Call SetSCRITSType("its_calc_SCRITS_WIM")
  If Not m_rst.EOF Then

    '-----
    'Weight-in-Motion (WIM)

    'Load Weigh in Motion Variables
    WIM_No = m_rst("WIM_No")
    WIM_Veh_I = m_rst("WIM_Veh_I")
    WIM_Veh_F = m_rst("WIM_Veh_F")
    WIM_AvgDelay = m_rst("WIM_AvgDelay")
    WIM_Per = m_rst("WIM_Per")

    'Traffic and Travel

    WIM_FWSav_I = WIM_No * WIM_Veh_I * WIM_AvgDelay * WIM_Per / 60
    WIM_FWSav_F = WIM_No * WIM_Veh_F * WIM_AvgDelay * WIM_Per / 60
    WIM_YrSav_I = WIM_FWSav_I * 365
    WIM_YrSav_F = WIM_FWSav_F * 365

    'Travel Time Calculations:
    'Yearly Travel Time Benefits in Minutes

```

```

WIM_TT_Min_I = WIM_YrSav_I * 60
WIM_TT_Min_F = WIM_YrSav_F * 60
m_WIM_TT_Min = (WIM_TT_Min_F - WIM_TT_Min_I) * N / 2
m_WIM_FrTT_Min = WIM_TT_Min

'Travel Time Benefits in Dollars
WIM_FrTT_Ben_I = (WIM_TT_Min_I) / 60 * Time_Value_Freight
WIM_FrTT_Ben_F = (WIM_TT_Min_F) / 60 * Time_Value_Freight
WIM_TTBen_Grad = (WIM_TT_Ben_F - WIM_TT_Ben_I) / N
m_WIM_TT_Ben = PofG * WIM_TTBen_Grad
m_WIM_TT_Ben = 0

'Operating Cost Calculations
'(No change in VMT)
m_WIM_User_Ben = 0
m_WIM_User_Transfer = 0

'Energy and Emissions
'(No change in VMT)
m_WIM_CO_Tons = 0
m_WIM_NOX_Tons = 0
m_WIM_PM10_Tons = 0
m_WIM_VOC_Tons = 0
m_WIM_Env_Ben = 0

'SAFETY Calculations
'Assumes safety benefits are negligible

m_WIM_Fatality = 0
m_WIM_Injury = 0
m_WIM_Property = 0

'Safety Benefit Calculations

m_WIM_Safety_Ben = 0

'Set module level values (to be displayed)
  m_Values.Add m_WIM_TT_Min, "WIM_TT_Min"
  m_Values.Add m_WIM_FrTT_Min, "WIM_FrTT_Min"
  m_Values.Add m_WIM_TT_Ben, "WIM_TT_Ben"
  m_Values.Add m_WIM_FrTT_Ben, "WIM_FrTT_Ben"

  m_Values.Add m_WIM_CO_Tons, "WIM_CO_Tons"
  m_Values.Add m_WIM_NOX_Tons, "WIM_NOX_Tons"
  m_Values.Add m_WIM_PM10_Tons, "WIM_PM10_Tons"
  m_Values.Add m_WIM_VOC_Tons, "WIM_Voc_Tons"
  m_Values.Add m_WIM_Env_Ben, "WIM_Env_Ben"

  m_Values.Add m_WIM_User_Ben, "WIM_User_Ben"
  m_Values.Add m_WIM_User_Transfer, "WIM_User_Transfer"

  m_Values.Add m_WIM_Fatality, "WIM_Fatality"
  m_Values.Add m_WIM_Injury, "WIM_Injury"
  m_Values.Add m_WIM_Property, "WIM_Property"
  m_Values.Add m_WIM_Safety_Ben, "WIM_Safety_Ben"
'Calls local function to set cost nodes
CalculateCosts "WIM", Discount_Rate, N

End If
End If

If bWIM Then

```

```
Call SetSCRITSType("its_calc_SCRITS_RR")
If Not m_rst.EOF Then
```

```
-----
'Railroad Crossing ITS
```

```
'Load Railroad Crossing Variables
RR_No = m_rst("RR_No")
RR_AccNo = m_rst("RR_AccNo")
RR_PerAcc = m_rst("RR_PerAcc")
```

```
'No travel time, user, or environmental benefits
```

```
m_RR_TT_Min = 0
m_RR_FrTT_Min = 0
m_RR_TT_Ben = 0
m_RR_FrTT_Ben = 0
m_RR_CO_Tons = 0
m_RR_NOX_Tons = 0
m_RR_PM10_Tons = 0
m_RR_VOC_Tons = 0
m_RR_Env_Ben = 0
m_RR_User_Ben = 0
m_RR_VOC_Tons = 0
m_RR_User_Transfer = 0
```

```
'Safety Benefit Calculations
```

```
RR_AccTot = RR_No * RR_AccNo
RR_AccRed = RR_AccTot * RR_PerAcc
```

'Assume that railroad accidents are likely very severe so accident damage likely to be similar to fatality damage costs.

```
m_RR_Fatality = RR_AccRed * N
m_RR_Injury = 0
m_RR_Property = 0
RR_NPVF_Safety_Ben = (Exp(-Discount_Rate * N) - 1) / (-Discount_Rate)
m_RR_Safety_Ben = RR_AccRed * Fatality_Cost * RR_NPVF_Safety_Ben
```

```
'Set module level values (to be displayed)
```

```
m_Values.Add m_RR_TT_Min, "RR_TT_Min"
m_Values.Add m_RR_FrTT_Min, "RR_FrTT_Min"
m_Values.Add m_RR_TT_Ben, "RR_TT_Ben"
m_Values.Add m_RR_FrTT_Ben, "RR_FrTT_Ben"

m_Values.Add m_RR_CO_Tons, "RR_CO_Tons"
m_Values.Add m_RR_NOX_Tons, "RR_NOX_Tons"
m_Values.Add m_RR_PM10_Tons, "Rm_R_PM10_Tons"
m_Values.Add m_RR_VOC_Tons, "RR_Voc_Tons"
m_Values.Add m_RR_Env_Ben, "RR_Env_Ben"

m_Values.Add m_RR_User_Ben, "RR_User_Ben"
m_Values.Add m_RR_User_Transfer, "RR_User_Transfer"

m_Values.Add m_RR_Fatality, "RR_Fatality"
m_Values.Add m_RR_Injury, "RR_Injury"
m_Values.Add m_RR_Property, "RR_Property"
m_Values.Add m_RR_Safety_Ben, "RR_Safety_Ben"
```

```
'Calls local function to set cost nodes
CalculateCosts "RR", Discount_Rate, N
```

End If
End If

If bTSS Then

Call SetSCRITSType("its_calc_SCRITS_TSS")

If Not m_rst.EOF Then

```
'-----
'Traffic Signal Systems (TSS)

'Load Traffic Signal Systems (TSS) Variables
TSS_PerImp = m_rst("TSS_PerImp")
TSS_Stops = m_rst("TSS_Stops")
TSS_PerRed = m_rst("TSS_PerRed")

'Traffic and Travel
TSS_AvgSpdWO_I = ArtVMT_BCI / RecArtVHT_BCI
TSS_AvgSpdWO_F = ArtVMT_BCF / RecArtVHT_BCF
TSS_AvgSpdW_I = TSS_AvgSpdWO_I + (TSS_AvgSpdWO_I * TSS_PerImp)
TSS_AvgSpdW_F = TSS_AvgSpdWO_F + (TSS_AvgSpdWO_F * TSS_PerImp)
TSS_WDVHTW_I = ArtVMT_BCI / TSS_AvgSpdW_I
TSS_WDVHTW_F = ArtVMT_BCF / TSS_AvgSpdW_F
TSS_WDSav_I = RecArtVHT_BCI - TSS_WDVHTW_I
TSS_WDSav_F = RecArtVHT_BCF - TSS_WDVHTW_F
TSS_YrSav_I = TSS_WDSav_I * Ann_Daily_Benefit
TSS_YrSav_F = TSS_WDSav_F * Ann_Daily_Benefit
TSS_WDStp_I = ArtVMT_BCI * TSS_Stops
TSS_WDStp_F = ArtVMT_BCF * TSS_Stops
TSS_YrStp_I = TSS_WDStp_I * TSS_PerRed
TSS_YrStp_F = TSS_WDStp_F * TSS_PerRed

'Travel Time Calculations:
'Yearly Travel Time Benefits in Minutes
TSS_TT_Min_I = (TSS_YrSav_I * 60)
TSS_TT_Min_F = TSS_YrSav_F * 60
m_TSS_TT_Min = (TSS_TT_Min_F - TSS_TT_Min_I) * N / 2
m_TSS_FrTT_Min = 0

'Travel Time Benefits in Dollars
TSS_TT_Ben_I = (TSS_TT_Min_I) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
TSS_TT_Ben_F = (TSS_TT_Min_F) / 60 * (Percent_Time_InVeh * Time_Value_Veh)
TSS_TTBen_Grad = (TSS_TT_Ben_F - TSS_TT_Ben_I) / N
m_TSS_TT_Ben = PofG * TSS_TTBen_Grad
m_TSS_FrTT_Ben = 0

'Operating Cost Calculations
'(No change in VMT)
m_TSS_User_Ben = 0
m_TSS_User_Transfer = 0

'Energy and Emissions
'(Does not currently differentiate between autos and freight vehicles.)
'CO Calculations
TSS_CORateWO_BCI = m_ikup_Pollution.RetrieveRate(TSS_AvgSpdWO_I, em_CO, veht_auto)
TSS_CORateW_BCI = m_ikup_Pollution.RetrieveRate(TSS_AvgSpdW_I, em_CO, veht_auto)
TSS_CORateDif_BCI = TSS_CORateW_BCI - TSS_CORateWO_BCI
TSS_CORateWO_BCF = m_ikup_Pollution.RetrieveRate(TSS_AvgSpdWO_F, em_CO, veht_auto)
TSS_CORateW_BCF = m_ikup_Pollution.RetrieveRate(TSS_AvgSpdW_F, em_CO, veht_auto)
TSS_CORateDif_BCF = TSS_CORateW_BCF - TSS_CORateWO_BCF
TSS_CO_Gram_BCI = -(TSS_CORateDif_BCI * ArtVMT_BCI)
TSS_CO_Gram_BCF = -(TSS_CORateDif_BCF * ArtVMT_BCF)
TSS_CO_Gram = (TSS_CO_Gram_BCI + TSS_CO_Gram_BCF) / 2 * N
```


$m_TSS_CO_Tons = TSS_CO_Gram / 907184.74$

'NOX Calculations

$TSS_NOXRateWO_BCI = m_Ikup_Pollution.RetrieveRate(TSS_AvgSpdWO_I, em_NOX, veht_auto)$
 $TSS_NOXRateW_BCI = m_Ikup_Pollution.RetrieveRate(TSS_AvgSpdW_I, em_NOX, veht_auto)$
 $TSS_NOXRateDif_BCI = TSS_NOXRateW_BCI - TSS_NOXRateWO_BCI$
 $TSS_NOXRateWO_BCF = m_Ikup_Pollution.RetrieveRate(TSS_AvgSpdWO_F, em_NOX, veht_auto)$
 $TSS_NOXRateW_BCF = m_Ikup_Pollution.RetrieveRate(TSS_AvgSpdW_F, em_NOX, veht_auto)$
 $TSS_NOXRateDif_BCF = TSS_NOXRateW_BCF - TSS_NOXRateWO_BCF$
 $TSS_NOX_Gram_BCI = (TSS_NOXRateDif_BCI * ArtVMT_BCI)$
 $TSS_NOX_Gram_BCF = (TSS_NOXRateDif_BCF * ArtVMT_BCF)$
 $TSS_NOX_Gram = (TSS_NOX_Gram_BCI + TSS_NOX_Gram_BCF) / 2 * N$
 $m_TSS_NOX_Tons = TSS_NOX_Gram / 907184.74$

'PM10 Calculations (No change in VMT)

$m_TSS_PM10_Tons = 0$

'VOC Calculations (No change in VMT)

$m_TSS_VOC_Tons = 0$

'Emissions Benefit Calculations

'NEED TO FIX

$CCTV_CO_Grad = (CCTV_CO_Gram_BCF - CCTV_CO_Gram_BCI) / N$
 $TSS_CO_Ben = TSS_CO_Grad * PofG * COTon_Cost / 907184.74$

$TSS_NOX_Grad = (TSS_NOX_Gram_BCF - TSS_NOX_Gram_BCI) / N$
 $TSS_NOX_Ben = TSS_NOX_Grad * PofG * NOXTon_Cost / 907184.74$

$TSS_PM10_Ben = 0$

$TSS_VOC_Ben = 0$

$m_TSS_Env_Ben = TSS_CO_Ben + TSS_NOX_Ben + TSS_PM10_Ben + TSS_VOC_Ben$

'Safety Calculations (SCRITS assumes accident reduction equal to stop percentage reduction)

$TSS_Fatality_I = ArtVMT_BCI * TSS_PerRed * Fat_Rate_Auto / 100000000$
 $TSS_Fatality_F = ArtVMT_BCF * TSS_PerRed * Fat_Rate_Auto / 100000000$
 $TSS_Injury_I = ArtVMT_BCI * TSS_PerRed * Inj_Rate_Auto / 100000000$
 $TSS_Injury_F = ArtVMT_BCF * TSS_PerRed * Inj_Rate_Auto / 100000000$
 $TSS_Property_I = ArtVMT_BCI * TSS_PerRed * Prop_Rate_Auto / 100000000$
 $TSS_Property_F = ArtVMT_BCF * TSS_PerRed * Prop_Rate_Auto / 100000000$
 $m_TSS_Fatality = (TSS_Fatality_F - TSS_Fatality_I) * N / 2$
 $m_TSS_Injury = (TSS_Injury_F - TSS_Injury_I) * N / 2$
 $m_TSS_Property = (TSS_Property_F - TSS_Property_I) * N / 2$

'Safety Benefit Calculations

$TSS_FatBen_Grad = (TSS_Fatality_F - TSS_Fatality_I) / N$
 $TSS_Fat_Ben = PofG * Fatality_Cost * TSS_FatBen_Grad$
 $TSS_InjBen_Grad = (TSS_Injury_F - TSS_Injury_I) / N$
 $TSS_Inj_Ben = PofG * Evident_Cost * TSS_InjBen_Grad$
 $TSS_PropBen_Grad = (TSS_Property_F - TSS_Property_I) / N$
 $TSS_Prop_Ben = PofG * PDO_Cost * TSS_PropBen_Grad$
 $m_TSS_Safety_Ben = TSS_Fat_Ben + TSS_Inj_Ben + TSS_Prop_Ben$

'Set module level values (to be displayed)

$m_Values.Add m_TSS_TT_Min, "TSS_TT_Min"$
 $m_Values.Add m_TSS_FrTT_Min, "TSS_FrTT_Min"$
 $m_Values.Add m_TSS_TT_Ben, "TSS_TT_Ben"$
 $m_Values.Add m_TSS_FrTT_Ben, "TSS_FrTT_Ben"$

```

    m_Values.Add m_TSS_CO_Tons, "TSS_CO_Tons"
    m_Values.Add m_TSS_NOX_Tons, "TSS_NOX_Tons"
    m_Values.Add m_TSS_PM10_Tons, "TSS_PM10_Tons"
    m_Values.Add m_TSS_VOC_Tons, "TSS_Voc_Tons"
    m_Values.Add m_TSS_Env_Ben, "TSS_Env_Ben"

    m_Values.Add m_TSS_User_Ben, "TSS_User_Ben"
    m_Values.Add m_TSS_User_Transfer, "TSS_User_Transfer"

    m_Values.Add m_TSS_Fatality, "TSS_Fatality"
    m_Values.Add m_TSS_Injury, "TSS_Injury"
    m_Values.Add m_TSS_Property, "TSS_Property"
    m_Values.Add m_TSS_Safety_Ben, "TSS_Safety_Ben"

    'Calls local function to set cost nodes
    CalculateCosts "TSS", Discount_Rate, N

    End If
End If

'-----
'Total Project Calculations

m_TT_Min = m_Values.RetrieveSum("TT_Min")
m_FrTT_Min = m_Values.RetrieveSum("FrTT_Min")
m_TT_Ben = m_Values.RetrieveSum("TT_Ben")
m_FrTT_Ben = m_Values.RetrieveSum("FrTT_Ben")
m_User_Ben = m_Values.RetrieveSum("User_Ben")
m_User_Transfer = m_Values.RetrieveSum("User_Transfer")
m_CO_Tons = m_Values.RetrieveSum("CO_Tons")
m_NOX_Tons = m_Values.RetrieveSum("NOX_Tons")
m_PM10_Tons = m_Values.RetrieveSum("PM10_Tons")
m_VOC_Tons = m_Values.RetrieveSum("VOC_Tons")
m_Env_Ben = m_Values.RetrieveSum("Env_Ben")
m_Fatality = m_Values.RetrieveSum("Fatality")
m_Injury = m_Values.RetrieveSum("Injury")
m_Property = m_Values.RetrieveSum("Property")
m_Safety_Ben = m_Values.RetrieveSum("Safety_Ben")

'Set module level values (to be displayed)
    m_Values.Add m_TT_Min, "TT_Min"
    m_Values.Add m_FrTT_Min, "FrTT_Min"
    m_Values.Add m_TT_Ben, "TT_Ben"
    m_Values.Add m_FrTT_Ben, "FrTT_Ben"

    m_Values.Add m_CO_Tons, "CO_Tons"
    m_Values.Add m_NOX_Tons, "NOX_Tons"
    m_Values.Add m_PM10_Tons, "PM10_Tons"
    m_Values.Add m_VOC_Tons, "Voc_Tons"
    m_Values.Add m_Env_Ben, "Env_Ben"

    m_Values.Add m_User_Ben, "User_Ben"
    m_Values.Add m_User_Transfer, "User_Transfer"

    m_Values.Add m_Fatality, "Fatality"
    m_Values.Add m_Injury, "Injury"
    m_Values.Add m_Property, "Property"
    m_Values.Add m_Safety_Ben, "Safety_Ben"

'Total Cost Calculations
    m_Total_Cost = m_Values.RetrieveSum("Total_Cost")
    m_WSDOT_Cost = m_Values.RetrieveSum("WSDOT_Cost")

```

```

m_Federal_Cost = m_Values.RetrieveSum("Federal_Cost")
m_Terminal_Cost = m_Values.RetrieveSum("Terminal_Cost")

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben
m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)

'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "Total_Cost"
m_Values.Add m_WSDOT_Cost, "WSDOT_Cost"
m_Values.Add m_Federal_Cost, "Federal_Cost"
m_Values.Add m_Terminal_Cost, "Terminal_Cost"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

'Update the rst Value
Call SetSCRITSType("its_calc_SCRITS")
Call UpdateRstField(m_rst, "WSDOT_TotalCost", m_WSDOT_Cost)

'Clean up lookup functions
Set m_lkup_Pollution = Nothing
Set m_lkup_Fatality = Nothing
Set m_lkup_Shield = Nothing

End Sub

Sub CalculateCosts(sType As String, DiscRate As Variant, ForePrd As Variant)

'Calculate economic analysis factors
PofA = (1 - (1 + DiscRate) ^ (-ForePrd)) / DiscRate
PofF = (1 + DiscRate) ^ (-ForePrd)

'Cost Calculations
'Capital Cost
For i = 1 To 5
    WSDOT_Cap = WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * 1 / ((1 + DiscRate) ^ (2 * i - 1)))
    Federal_Cap = Federal_Cap + (m_rst("FederalCap_Bi" & i) * 1 / ((1 + DiscRate) ^ (2 * i - 1)))
    Other1_Cap = Other1_Cap + (m_rst("Other1Cap_Bi" & i) * 1 / ((1 + DiscRate) ^ (2 * i - 1)))
    Other2_Cap = Other2_Cap + (m_rst("Other2Cap_Bi" & i) * 1 / ((1 + DiscRate) ^ (2 * i - 1)))
    Other3_Cap = Other3_Cap + (m_rst("Other3Cap_Bi" & i) * 1 / ((1 + DiscRate) ^ (2 * i - 1)))
Next
Cap_Cost = WSDOT_Cap + Federal_Cap + Other1_Cap + Other2_Cap + Other3_Cap

'Operations and Maintenance Cost
WSDOT_OM = m_rst("WSDOT_annOM") * PofA
Federal_OM = m_rst("Federal_annOM") * PofA
Other1_OM = m_rst("Other1_annOM") * PofA
Other2_OM = m_rst("Other2_annOM") * PofA
Other3_OM = m_rst("Other3_annOM") * PofA
OpMaint_Cost = WSDOT_OM + Federal_OM + Other1_OM + Other2_OM + Other3_OM

'Terminal cost
Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
WSDOT_Cost = WSDOT_Cap + WSDOT_OM
Federal_Cost = Federal_Cap + Federal_OM
Other1_Cost = Other1_Cap + Other1_OM
Other2_Cost = Other2_Cap + Other2_OM
Other3_Cost = Other3_Cap + Other3_OM

```

```
Total_Cost = WSDOT_Cost + Federal_Cost + Other1_Cost + Other2_Cost + Other3_Cost
```

```
'Environmental Retrofit Calculations
```

```
'Environmental Retrofit Costs not applicable
```

```
'Benefit-Cost Calculations
```

```
TT_Ben = m_Values.Retrieve(sType & "_TT_Ben")
```

```
User_Ben = m_Values.Retrieve(sType & "_User_Ben")
```

```
Env_Ben = m_Values.Retrieve(sType & "_Env_Ben")
```

```
Safety_Ben = m_Values.Retrieve(sType & "_Safety_Ben")
```

```
Total_Benefit = TT_Ben + User_Ben + Env_Ben + Safety_Ben
```

```
BCR = Total_Benefit / (Total_Cost - Terminal_Cost)
```

```
WSDOT_BCR = Total_Benefit / (WSDOT_Cost - Terminal_Cost)
```

```
'Set module level values (to be displayed)
```

```
  m_Values.Add Total_Cost, sType & "_Total_Cost"
```

```
  m_Values.Add WSDOT_Cost, sType & "_WSDOT_Cost"
```

```
  m_Values.Add Federal_Cost, sType & "_Federal_Cost"
```

```
  m_Values.Add Terminal_Cost, sType & "_Terminal_Cost"
```

```
  m_Values.Add BCR, sType & "_BCR"
```

```
  m_Values.Add WSDOT_BCR, sType & "_WSDOT_BCR"
```

```
End Sub
```

```
Private Sub CalculateOutObjScores()
```

```
On Error Resume Next
```

```
'Calculation for the System Operation and Maintenance
```

```
  m_Sys_OM = (m_rst("Q1A") * 34) + (33 * m_rst("Q1B")) + (33 * m_rst("Q1D"))
```

```
  m_Values.Add m_Sys_OM, "SYS_OM"
```

```
'Calculation for the System Preservation
```

```
  m_Sys_Pres = 50 * m_rst("Q2A")
```

```
  m_Values.Add m_Sys_Pres, "SYS_PRES"
```

```
'Calculation for the Special Needs Transportation
```

```
  m_Sp_Needs = 75 * m_rst("Q3A")
```

```
  m_Values.Add m_Sp_Needs, "SP_NEEDS"
```

```
'Calculation for the Congestion Relief
```

```
  If m_rst("WTP_Corridor") Then
```

```
    If TT_Min > 0 Then
```

```
      m_Cong_Rel = 100
```

```
    ElseIf m_TT_Min = 0 Then
```

```
      m_Cong_Rel = 50
```

```
    Else
```

```
      m_Cong_Rel = 0
```

```
    End If
```

```
  Else
```

```
    m_Cong_Rel = 0
```

```
  End If
```

```
  m_Values.Add m_Cong_Rel, "CONG_REL"
```

```
'Calculation for Increased Travel Options
```

```
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
```

```
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
```

```
'Calculation for Seamless Connections
```

```
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
```

```
  m_Values.Add m_Seamless, "SEAMLESS"
```

```
'Calculation for Safety
```

```
  If m_Safety_Ben > 0 Then
```

```
    SafetyA = 80
```

```
  ElseIf m_Safety_Ben = 0 Then
```

```
    SafetyA = 40
```

```
  Else
```

```
    SafetyA = 0
```

```
  End If
```

```

    m_Safety = SafetyA + (20 * m_rst("Q7B"))
    m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
    m_Commnty = -((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E")))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") > 0 Then
        m_Collab = 50
    Else
        m_Collab = 0
    End If
    m_Collab = m_Collab + (50 * m_rst("Q10A"))
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    St_Freight = m_rst("St_Freight")
    If St_Freight < 6 And m_FrTT_Min > 0 Then
        FreightA = 45 + 45 * ((6 - St_Freight) / 5)
    ElseIf St_Freight < 6 And m_FrTT_Min = 0 Then
        FreightA = 45
    ElseIf St_Freight < 6 And m_FrTT_Min < 0 Then
        FreightA = 45 - 45 * ((6 - St_Freight) / 5)
    Else
        FreightA = 0
    End If
    m_Freight = FreightA + (10 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    Air_Quality = m_rst("Air_Quality")
    If m_Env_Ben > 0 Then
        If Air_Quality = 3 Then
            AirQualA = 80
        ElseIf Air_Quality = 2 Then
            AirQualA = 60
        Else
            AirQualA = 40
        End If
    ElseIf m_Env_Ben = 0 Then
        If Air_Quality = 3 Then
            AirQualA = 70
        ElseIf Air_Quality = 2 Then
            AirQualA = 50
        Else
            AirQualA = 40
        End If
    Else
        If Air_Quality = 3 Then
            AirQualA = 0
        ElseIf Air_Quality = 2 Then
            AirQualA = 20
        Else
            AirQualA = 40
        End If
    End If

```

```

    End If
  End If
  m_Air_Qual = AirQualA + (10 * m_rst("Q14A")) + (10 * m_rst("Q14B"))
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (17 * m_rst("Q15A")) + (16 * m_rst("Q15B")) + _
    (16 * m_rst("Q15C"))
  m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
  HabitatA = ((5 * m_rst("Q16A")) + (5 * m_rst("Q16B")) + _
    (5 * m_rst("Q16C")) + (5 * m_rst("Q16D")))
  If m_rst("Q16E") = 0 Then
    HabitatB = 40
  ElseIf m_rst("Q16E") = 1 Then
    HabitatB = 20
  ElseIf m_rst("Q16E") = 2 Then
    HabitatB = 10
  Else
    HabitatB = 0
  End If
  If m_rst("Q16F") = 0 Then
    HabitatC = 40
  ElseIf m_rst("Q16F") = 1 Then
    HabitatC = 20
  ElseIf m_rst("Q16F") = 2 Then
    HabitatC = 10
  Else
    HabitatC = 0
  End If
  m_Habitat = HabitatA + HabitatB + HabitatC
  m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
  m_Resource = 100 * m_rst("Q17")
  m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
  Dim rtnval

  rtnval = m_Values.Retrieve(itemname)

  GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
  a_rst.Close
  m_rst.Close
  m_dbs.Close
  Set a_rst = Nothing
  Set m_rst = Nothing
  Set m_dbs = Nothing
  Set m_Values = Nothing
End Sub

```

Program Code for Non-Motorized Projects

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim a_rst As DAO.Recordset

Dim m_qryName As String

Dim m_calcsComplete As Boolean

Dim m_Values As ValueCollection

'Variables used in intermediate calculations (variants)

Dim m_Ped_Shift

Dim m_Bike_Shift

Dim m_VMT_Shift

'Variables to hold the calculated values (variants)

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_User_Transfer

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Terminal_Cost

Dim m_BCR

Dim m_WSDOT_BCR

'Variables to hold the outcome objective scores (variants)

Dim m_Sys_OM

Dim m_Sys_Pres

Dim m_Sp_Needs

Dim m_Cong_Rel

Dim m_Trav_Opt

Dim m_Seamless

Dim m_Safety

Dim m_Security

Dim m_Commnty

Dim m_Collab

Dim m_Freight

Dim m_Econ_Pro

Dim m_Tourism

Dim m_Air_Qual

Dim m_Wtr_Qual

```

Dim m_Habitat
Dim m_Resource

'Speed Assumed for lookup table
Private Const AUTOSPEED = 35

Private Sub Class_Initialize()
    m_qryName = "nonmot_calc_All"
    m_calcsComplete = False
End Sub

Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean

    Dim atype As String
    Dim qryDef As DAO.QueryDef

    Set m_dbs = CurrentDb

    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid

    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then

        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection

        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If

        'Open up a assumption recordset - depend on if in MICA or not
        Set qryDef = m_dbs.QueryDefs(atype)
        qryDef.Parameters!asmptnID = pid

        ' Set recordset to new values.
        Set a_rst = qryDef.OpenRecordset

        'Calculate all of the values for the project type
        CalculateBenefits
        CalculateOutObjScores

        'Check to see if the calcs are done for this project and set input status accordingly
        If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
        m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

        Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

    End If

```



```
'Close the Querydef object
qryDef.Close
Set qryDef = Nothing
```

```
CalculateProjectType = m_calcsComplete
```

```
End Function
```

```
Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property
```

```
Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property
```

```
Private Sub CalculateBenefits()
```

```
    On Error Resume Next
```

```
    'Calculate Forecast Period
```

```
    Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")
```

```
    'Calculate economic analysis factors
```

```
    Discount_Rate = a_rst("Discount_Rate")
```

```
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
```

```
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / _
        ((1 + Discount_Rate) ^ Forecast_Period))
```

```
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)
```

```
    'Pedestrians
```

```
    BCI_Peds = m_rst("Ped_WkDy_BCI") * 260 + m_rst("Ped_WkEnd_BCI") * 105
```

```
    BCF_Peds = m_rst("Ped_WkDy_BCF") * 260 + m_rst("Ped_WkEnd_BCF") * 105
```

```
    PCI_Peds = m_rst("Ped_WkDy_PCI") * 260 + m_rst("Ped_WkEnd_PCI") * 105
```

```
    PCF_Peds = m_rst("Ped_WkDy_PCF") * 260 + m_rst("Ped_WkEnd_PCF") * 105
```

```
    Ped_Shift_i = PCI_Peds - BCI_Peds
```

```
    Ped_Shift_f = PCF_Peds - BCF_Peds
```

```
    m_Ped_Shift = (Ped_Shift_i + Ped_Shift_f) * Forecast_Period / 2
```

```
    'Bicycles
```

```
    BCI_Bikes = m_rst("Bike_WkDy_BCI") * 260 + m_rst("Bike_WkEnd_BCI") * 105
```

```
    BCF_Bikes = m_rst("Bike_WkDy_BCF") * 260 + m_rst("Bike_WkEnd_BCF") * 105
```

```
    PCI_Bikes = m_rst("Bike_WkDy_PCI") * 260 + m_rst("Bike_WkEnd_PCI") * 105
```

```
    PCF_Bikes = m_rst("Bike_WkDy_PCF") * 260 + m_rst("Bike_WkEnd_PCF") * 105
```

```
    Bike_Shift_i = PCI_Bikes - BCI_Bikes
```

```
    Bike_Shift_f = PCF_Bikes - BCF_Bikes
```

```
    m_Bike_Shift = (Bike_Shift_i + Bike_Shift_f) * Forecast_Period / 2
```

```
    'Automobile trips and VMT diverted
```

```
    Ped_Divert = a_rst("Walk_Divert")
```

```
    Bike_Divert = a_rst("Bike_Divert")
```

```
    avo = a_rst("avo")
```

```
    Length_Ped_Trip = m_rst("Length_Ped_Trip")
```

```
    Length_Bike_Trip = m_rst("Length_Bike_Trip")
```

```
    Vmt_shift_i = ((Ped_Shift_i * Ped_Divert * Length_Ped_Trip) + _
```

```

(Bike_Shift_i * Bike_Divert * Length_Bike_Trip)) / avo * (-1)
Vmt_shift_f = ((Ped_Shift_f * Ped_Divert * Length_Ped_Trip) + _
(Bike_Shift_f * Bike_Divert * Length_Bike_Trip)) / avo * (-1)
m_VMT_Shift = (Vmt_shift_i + Vmt_shift_f) * Forecast_Period / 2

```

```

'Set module level values (to be displayed)
m_Values.Add m_Ped_Shift, "PED_SHIFT"
m_Values.Add m_Bike_Shift, "BIKE_SHIFT"
m_Values.Add m_VMT_Shift, "VMT_SHIFT"

```

'Operating Cost Savings Calculations

```

Veh_OpCost_Full = a_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = a_rst("Veh_OpCost_Direct")
If a_rst("Full_Cost") Then
  OpCost_i = (-1) * Vmt_shift_i * Veh_OpCost_Full
  OpCost_f = (-1) * Vmt_shift_f * Veh_OpCost_Full
Else
  OpCost_i = (-1) * Vmt_shift_i * Veh_OpCost_Direct
  OpCost_f = (-1) * Vmt_shift_f * Veh_OpCost_Direct
End If
'User Benefit NPV Calculation
G_Opcost = (OpCost_f - OpCost_i) / (Forecast_Period - 1)
m_User_Ben = (OpCost_i * PofA) + (G_Opcost * PofG)

```

'Travel Time Savings Calculations

```

'No travel time benefits are calculated for this project type
m_TT_Ben = 0

```

```

'Set module level values (to be displayed)
m_Values.Add m_Approach, "APPROACH"
m_Values.Add m_User_Ben, "USER_BEN"
m_Values.Add m_TT_Ben, "TT_BEN"

```

'Air Pollution Calculations - for diverted auto trips

```

CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)

If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If
COTon_Cost = a_rst("COTon_Cost")
VOC_Rate_Auto = a_rst("VOC_Rate_Auto")
VOCTon_Cost = a_rst("VOCTon_Cost")
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
NOXTon_Cost = a_rst("NOXTon_Cost")
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")
PM10Ton_Cost = a_rst("PM10Ton_Cost")

```

```

CO_Tons_i = ((Vmt_shift_i * CO_Rate_Auto)) * (1 / 1000) * (0.9842 / 1000)
CO_Tons_f = ((Vmt_shift_f * CO_Rate_Auto)) * (1 / 1000) * (0.9842 / 1000)
VOC_Tons_i = ((Vmt_shift_i * VOC_Rate_Auto)) * (1 / 1000) * (0.9842 / 1000)
VOC_Tons_f = ((Vmt_shift_f * VOC_Rate_Auto)) * (1 / 1000) * (0.9842 / 1000)
NOX_Tons_i = ((Vmt_shift_i * NOX_Rate_Auto)) * (1 / 1000) * (0.9842 / 1000)
NOX_Tons_f = ((Vmt_shift_f * NOX_Rate_Auto)) * (1 / 1000) * (0.9842 / 1000)

```

```
PM10_Tons_i = ((Vmt_shift_i * PM10_Rate_Auto)) * (1 / 1000) * (0.9842 / 1000)
PM10_Tons_f = ((Vmt_shift_f * PM10_Rate_Auto)) * (1 / 1000) * (0.9842 / 1000)
```

```
'Emission Sums
```

```
  m_CO_Tons = (m_CO_Tons_f + m_CO_Tons_i) * Forecast_Period / 2
  m_VOC_Tons = (m_VOC_Tons_f + m_VOC_Tons_i) * Forecast_Period / 2
  m_NOX_Tons = (m_NOX_Tons_f + m_NOX_Tons_i) * Forecast_Period / 2
  m_PM10_Tons = (m_PM10_Tons_f + m_PM10_Tons_i) * Forecast_Period / 2
```

```
'Emission Benefit Calculations
```

```
EnvBen_i = (CO_Tons_i * COTon_Cost + VOC_Tons_i * VOCTon_Cost + _
  NOX_Tons_i * NOXTon_Cost + PM10_Tons_i * PM10Ton_Cost) * (-1)
EnvBen_f = (CO_Tons_f * COTon_Cost + VOC_Tons_f * VOCTon_Cost + _
  NOX_Tons_f * NOXTon_Cost + PM10_Tons_f * PM10Ton_Cost) * (-1)
G_EnvBen = (EnvBen_f - EnvBen_i) / (Forecast_Period - 1)
```

```
'Emission NPV Calculations
```

```
m_Env_Ben = (EnvBen_i * PofA) + (G_EnvBen * PofG)
```

```
'Set module level values (to be displayed)
```

```
  m_Values.Add m_CO_Tons, "CO_TONS"
  m_Values.Add m_VOC_Tons, "VOC_TONS"
  m_Values.Add m_NOX_Tons, "NOX_TONS"
  m_Values.Add m_PM10_Tons, "PM10_TONS"
  m_Values.Add m_Env_Ben, "ENV_BEN"
```

```
'Safety Calculations
```

```
Possible_Cost = a_rst("Possible_Cost")
Evident_Cost = a_rst("Evident_Cost")
Disable_Cost = a_rst("Disable_Cost")
Fatality_Cost = a_rst("Fatality_Cost")
PDO_Cost = a_rst("PDO_Cost")
```

```
'Accident reduction for facility users
```

```
acc_period = m_rst("end_acc_data") - m_rst("begin_acc_data")
If acc_period <> 0 Then
  ann_fac_property = m_rst("prev_no_inj") / acc_period
  ann_fac_poss_inj = m_rst("prev_poss_inj") / acc_period
  ann_fac_evdnt_inj = m_rst("prev_evdnt_inj") / acc_period
  ann_fac_disabl_inj = m_rst("prev_disabl_inj") / acc_period
  ann_fac_injury = ann_fac_poss_inj + ann_fac_evdnt_inj + ann_fac_disabl_inj
  ann_fac_fatality = m_rst("prev_fatality") / acc_period
End If
Safety_Facility = (ann_fac_property * PDO_Cost + ann_fac_poss_inj * Possible_Cost + _
  ann_fac_evdnt_inj * Evident_Cost + ann_fac_disabl_inj * Disable_Cost + _
  ann_fac_fatality * Fatality_Cost) * PofA
```

```
'Accident reduction due to diverted autos
```

```
'Look up accident rates from lookup table
Fat_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_auto)
If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
End If
Inj_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Injury, veht_auto)
If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
```

```

    Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
End If
Prop_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Property, veht_auto)
If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
End If

```

```

auto_Fatality_i = Vmt_shift_i * Fat_Rate_Auto / 1000000
auto_Fatality_f = Vmt_shift_f * Fat_Rate_Auto / 1000000
auto_Injury_i = Vmt_shift_i * Inj_Rate_Auto / 1000000
auto_Injury_f = Vmt_shift_f * Inj_Rate_Auto / 1000000
auto_Property_i = Vmt_shift_i * Prop_Rate_Auto / 1000000
auto_Property_f = Vmt_shift_f * Prop_Rate_Auto / 1000000
auto_Fatality = (auto_Fatality_i + auto_Fatality_f) * (Forecast_Period) / 2
auto_Injury = (auto_Injury_i + auto_Injury_f) * (Forecast_Period) / 2
auto_Property = (auto_Property_i + auto_Property_f) * (Forecast_Period) / 2

```

'Total Accident reduction

```

m_Property = ((-1) * ann_fac_property * Forecast_Period) + auto_Property
m_Injury = ((-1) * ann_fac_injury * Forecast_Period) + auto_Injury
m_Fatality = ((-1) * ann_fac_fatality * Forecast_Period) + auto_Fatality

```

'Calculate safety benefits

```

Safety_Autodivert_i = (auto_Fatality_i * Fatality_Cost + _
    auto_Injury_i * Evident_Cost + auto_Property_i * PDO_Cost) * (-1)
Safety_Autodivert_f = (auto_Fatality_f * Fatality_Cost + _
    auto_Injury_f * Evident_Cost + auto_Property_f * PDO_Cost) * (-1)
G_divert_safety = (Safety_Autodivert_f - Safety_Autodivert_i) / _
    (Forecast_Period - 1)
Safety_Divert = (Safety_Autodivert_i * PofA) + (G_divert_safety * PofG)

```

```

m_Safety_Ben = Safety_Facility + Safety_Divert

```

'Set module level values (to be displayed)

```

m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

```

'Cost Calculations

'Capital Cost

For i = 1 To 5

```

m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))

```

Next

```

m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

```

```

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
If (m_Total_Cost - m_Terminal_Cost) = 0 Then
m_BCR = 0
Else
m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
End If

```

```

If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
  m_WSDOT_BCR = 0
Else
  m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
End If
'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

```

Private Sub CalculateOutObjScores()
  On Error Resume Next
  'Calculation for the System Operation and Maintenance
  m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
  m_Values.Add m_Sys_OM, "SYS_OM"
  'Calculation for the System Preservation
  m_Sys_Pres = 100 * m_rst("Q2A")
  m_Values.Add m_Sys_Pres, "SYS_PRES"
  'Calculation for the Special Needs Transportation
  m_Sp_Needs = 100 * m_rst("Q3A")
  m_Values.Add m_Sp_Needs, "SP_NEEDS"
  'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
  End If

```

```

Else
  m_Cong_Rel = (m_rst("Q4") * 50)
End If
m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
  Else
    m_Safety = m_rst("Q7B") * 50
  End If
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  'The -1 Multiplication Corrects the negative sign introduced for true
  m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B"))) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E"))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
  Else
    m_Collab = (m_rst("Q10A") * 50) + 50
  End If
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
  If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + m_rst("Q14A") * 50
  Else
    m_Air_Qual = 0
  End If
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
    (33 * m_rst("Q15C"))
  m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
  m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B"))) + _
    (25 * m_rst("Q16C")) + (25 * m_rst("Q16D")) / _

```

```
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
    'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub
```


Program Code for Rail Projects

Freight Car Purchase

Option Compare Database

'Variables for the class level database objects

```
Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection
```

'Variables used in intermediate calculations (variants)

```
Dim Forecast_Period
Dim m_OpCost
Dim m_Pavement_Sav
Dim m_Truck_Rem
Dim m_VMT_Shift
Dim m_Rail_Vmt_Add
Dim m_Approach
Dim m_WSDOT_Cap
Dim m_Federal_Cap
Dim m_Other1_Cap
Dim m_Other2_Cap
Dim m_Other3_Cap
Dim m_WSDOT_OM
Dim m_Federal_OM
Dim m_Other1_OM
Dim m_Other2_OM
Dim m_Other3_OM
```

'Variables to hold the calculated values (variants)

```
Dim m_Total_Benefit
Dim m_TT_Ben
Dim m_User_Ben
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
Dim m_Other2_Cost
Dim m_Other3_Cost
Dim m_Cap_Cost
Dim m_OpMaint_Cost
```

```
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
'Speed Assumed for lookup table
Private Const TRUCKSPEED = 50
```

```
Private Sub Class_Initialize()
    m_qryName = "rail_calc_Freightcar"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
```

```

    pid = asmpnID
End If

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmpnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
    Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")

'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - _
        (Forecast_Period / ((1 + Discount_Rate) ^ Forecast_Period)))
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Shift from freight trucks to rail
    farm_tonpertruck = a_rst("farm_tonpertruck")
    farm_truckshift_i = (m_rst("farm_ton_PCI") - m_rst("farm_ton_BCI")) / farm_tonpertruck
    farm_truckshift_f = (m_rst("farm_ton_PCF") - m_rst("farm_ton_BCF")) / farm_tonpertruck
    farm_truckshift = (farm_truckshift_i + farm_truckshift_f) * Forecast_Period / 2

```

```

    lumber_tonpertruck = a_rst("lumber_tonpertruck")
    lumber_truckshift_i = (m_rst("lumber_ton_PCI") - m_rst("lumber_ton_BCI")) /
lumber_tonpertruck
    lumber_truckshift_f = (m_rst("lumber_ton_PCF") - m_rst("lumber_ton_BCF")) /
lumber_tonpertruck
    lumber_truckshift = (lumber_truckshift_i + lumber_truckshift_f) * Forecast_Period / 2
    mixed_tonpertruck = a_rst("mixed_tonpertruck")
    mixed_truckshift_i = (m_rst("mixed_ton_PCI") - m_rst("mixed_ton_BCI")) /
mixed_tonpertruck
    mixed_truckshift_f = (m_rst("mixed_ton_PCF") - m_rst("mixed_ton_BCF")) /
mixed_tonpertruck
    mixed_truckshift = (mixed_truckshift_i + mixed_truckshift_f) * Forecast_Period / 2
    chemical_tonpertruck = a_rst("chemical_tonpertruck")
    chemical_truckshift_i = (m_rst("chemical_ton_PCI") - m_rst("chemical_ton_BCI")) /
chemical_tonpertruck
    chemical_truckshift_f = (m_rst("chemical_ton_PCF") - m_rst("chemical_ton_BCF")) /
chemical_tonpertruck
    chemical_truckshift = (chemical_truckshift_i + chemical_truckshift_f) * Forecast_Period / 2
    food_tonpertruck = a_rst("food_tonpertruck")
    food_truckshift_i = (m_rst("food_ton_PCI") - m_rst("food_ton_BCI")) / food_tonpertruck
    food_truckshift_f = (m_rst("food_ton_PCF") - m_rst("food_ton_BCF")) / food_tonpertruck
    food_truckshift = (food_truckshift_i + food_truckshift_f) * Forecast_Period / 2
    paper_tonpertruck = a_rst("paper_tonpertruck")
    paper_truckshift_i = (m_rst("paper_ton_PCI") - m_rst("paper_ton_BCI")) /
paper_tonpertruck
    paper_truckshift_f = (m_rst("paper_ton_PCF") - m_rst("paper_ton_BCF")) /
paper_tonpertruck
    paper_truckshift = (paper_truckshift_i + paper_truckshift_f) * Forecast_Period / 2
    petrol_tonpertruck = a_rst("petrol_tonpertruck")
    petrol_truckshift_i = (m_rst("petrol_ton_PCI") - m_rst("petrol_ton_BCI")) /
petrol_tonpertruck
    petrol_truckshift_f = (m_rst("petrol_ton_PCF") - m_rst("petrol_ton_BCF")) /
petrol_tonpertruck
    petrol_truckshift = (petrol_truckshift_i + petrol_truckshift_f) * Forecast_Period / 2
    scrap_tonpertruck = a_rst("scrap_tonpertruck")
    scrap_truckshift_i = (m_rst("scrap_ton_PCI") - m_rst("scrap_ton_BCI")) /
scrap_tonpertruck
    scrap_truckshift_f = (m_rst("scrap_ton_PCF") - m_rst("scrap_ton_BCF")) /
scrap_tonpertruck
    scrap_truckshift = (scrap_truckshift_i + scrap_truckshift_f) * Forecast_Period / 2
    stone_tonpertruck = a_rst("stone_tonpertruck")
    stone_truckshift_i = (m_rst("stone_ton_PCI") - m_rst("stone_ton_BCI")) /
stone_tonpertruck
    stone_truckshift_f = (m_rst("stone_ton_PCF") - m_rst("stone_ton_BCF")) /
stone_tonpertruck
    stone_truckshift = (stone_truckshift_i + stone_truckshift_f) * Forecast_Period / 2
    metal_tonpertruck = a_rst("metal_tonpertruck")
    metal_truckshift_i = (m_rst("metal_ton_PCI") - m_rst("metal_ton_BCI")) /
metal_tonpertruck
    metal_truckshift_f = (m_rst("metal_ton_PCF") - m_rst("metal_ton_BCF")) /
metal_tonpertruck
    metal_truckshift = (metal_truckshift_i + metal_truckshift_f) * Forecast_Period / 2
    trans equip_tonpertruck = a_rst("trans equip_tonpertruck")
    trans equip_truckshift_i = (m_rst("trans equip_ton_PCI") - m_rst("trans equip_ton_BCI"))
/ trans equip_tonpertruck

```

```

trans_equip_truckshift_f = (m_rst("trans_equip_ton_PCF") -
m_rst("trans_equip_ton_BCF")) / trans_equip_tonpertruck
trans_equip_truckshift = (trans_equip_truckshift_i + trans_equip_truckshift_f) *
Forecast_Period / 2

```

```

truck_rem_i = farm_truckshift_i + lumber_truckshift_i + mixed_truckshift_i +
chenical_truckshift_i + food_truckshift_i + paper_truckshift_i + petrol_truckshift_i +
scrap_truckshift_i + stone_truckshift_i + metal_truckshift_i + trans_equip_truckshift_i

```

```

truck_rem_f = farm_truckshift_f + lumber_truckshift_f + mixed_truckshift_f +
chenical_truckshift_f + food_truckshift_f + paper_truckshift_f + petrol_truckshift_f +
scrap_truckshift_f + stone_truckshift_f + metal_truckshift_f + trans_equip_truckshift_f

```

```

m_Truck_Rem = (truck_rem_i + truck_rem_f) * Forecast_Period / 2

```

```

'Truck VMT shift

```

```

freight_trip_length_BC = m_rst("freight_trip_length_BC")
freight_trip_length_PC = m_rst("freight_trip_length_PC")

```

```

truck_vmt_shift_i = (-1) * truck_rem_i * freight_trip_length_PC
truck_vmt_shift_f = (-1) * truck_rem_f * freight_trip_length_PC
m_VMT_Shift = (truck_vmt_shift_i + truck_vmt_shift_f) * Forecast_Period / 2

```

```

'Added Rail VMT

```

```

rail_vmt_add_i = ((m_rst("freight_trains_wk_PCI") * freight_trip_length_PC) - _
(m_rst("freight_trains_wk_BCI") * freight_trip_length_BC)) * 52
rail_vmt_add_f = ((m_rst("freight_trains_wk_PCF") * freight_trip_length_PC) - _
(m_rst("freight_trains_wk_BCF") * freight_trip_length_BC)) * 52
m_Rail_Vmt_Add = (rail_vmt_add_i + rail_vmt_add_f) * Forecast_Period / 2

```

```

'Set module level values (to be displayed)

```

```

m_Values.Add Forecast_Period, "FORECAST_PERIOD"
m_Values.Add m_Truck_Rem, "TRUCK_REM"
m_Values.Add m_VMT_Shift, "VMT_SHIFT"
m_Values.Add m_Rail_Vmt_Add, "RAIL_VMT_ADD"

```

```

'Travel Time Savings Calculations

```

```

'No travel time savings calculated for this project
m_TT_Ben = 0

```

```

'Operating Cost Savings Calculations

```

```

Truck_OpCost_Full = a_rst("Truck_OpCost_Full")
Truck_OpCost_Direct = a_rst("Truck_OpCost_Direct")
If a_rst("Full_Cost") Then
    m_Approach = "Full cost"
    OpCost_i = (-1) * truck_vmt_shift_i * Truck_OpCost_Full
    OpCost_f = (-1) * truck_vmt_shift_f * Truck_OpCost_Full
Else
    m_Approach = "Direct cost"
    OpCost_i = (-1) * truck_vmt_shift_i * Truck_OpCost_Direct
    OpCost_f = (-1) * truck_vmt_shift_f * Truck_OpCost_Direct
End If

```

```

G_Opcost = (OpCost_f - OpCost_i) / (Forecast_Period - 1)
m_OpCost = (OpCost_i * PofA) + (G_Opcost * PofG)

```

'Pavement Cost Savings

```
truck_pave_cost = a_rst("truck_pave_cost")
Pavement_Sav_i = (-1) * truck_vmt_shift_i * truck_pave_cost
Pavement_Sav_f = (-1) * truck_vmt_shift_f * truck_pave_cost
```

```
G_Pavement_Sav = (Pavement_Sav_f - Pavement_Sav_i) / (Forecast_Period - 1)
m_Pavement_Sav = (Pavement_Sav_i * PofA) + (G_Pavement_Sav * PofG)
```

'User Benefit NPV Calculation

```
m_User_Ben = m_OpCost + m_Pavement_Sav
```

'Travel Time Savings Calculations

```
'No travel time benefits are calculated for this project type
m_TT_Ben = 0
```

'Set module level values (to be displayed)

```
m_Values.Add m_Approach, "APPROACH"
m_Values.Add m_OpCost, "OPCOST"
m_Values.Add m_Pavement_Sav, "PAVEMENT_SAV"
m_Values.Add m_User_Ben, "USER_BEN"
m_Values.Add m_TT_Ben, "TT_BEN"
```

'Air Pollution Calculations - for diverted auto trips

```
CO_Rate_Truck = LookupAirPollution(m_dbs, TRUCKSPEED, em_CO, veht_Truck)
```

```
If a_rst("CO_Rate_Truck") <> CO_Rate_Truck Or IsNull(a_rst("CO_Rate_Truck")) Then
    Call UpdateRstField(a_rst, "CO_Rate_Truck", CO_Rate_Truck)
```

```
End If
```

```
CO_Rate_Rail = a_rst("CO_Rate_Rail")
```

```
COTon_Cost = a_rst("COTon_Cost")
```

```
VOC_Rate_Truck = a_rst("VOC_Rate_Truck")
```

```
VOC_Rate_Rail = a_rst("VOC_Rate_Rail")
```

```
VOCTon_Cost = a_rst("VOCTon_Cost")
```

```
NOX_Rate_Truck = LookupAirPollution(m_dbs, TRUCKSPEED, em_NOX, veht_Truck)
```

```
If a_rst("NOX_Rate_Truck") <> NOX_Rate_Truck Or IsNull(a_rst("NOX_Rate_Truck")) Then
    Call UpdateRstField(a_rst, "NOX_Rate_Truck", NOX_Rate_Truck)
```

```
End If
```

```
NOX_Rate_Rail = a_rst("NOX_Rate_Rail")
```

```
NOXTon_Cost = a_rst("NOXTon_Cost")
```

```
PM10_Rate_Truck = a_rst("PM10_Rate_Truck")
```

```
PM10_Rate_Rail = a_rst("PM10_Rate_Rail")
```

```
PM10Ton_Cost = a_rst("PM10Ton_Cost")
```

```
CO_Tons_i = ((truck_vmt_shift_i * CO_Rate_Truck) - _
    (rail_vmt_add_i * CO_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
CO_Tons_f = ((truck_vmt_shift_f * CO_Rate_Truck) - _
    (rail_vmt_add_f * CO_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
VOC_Tons_i = ((truck_vmt_shift_i * VOC_Rate_Truck) - _
    (rail_vmt_add_i * VOC_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
VOC_Tons_f = ((truck_vmt_shift_f * VOC_Rate_Truck) - _
    (rail_vmt_add_f * VOC_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
NOX_Tons_i = ((truck_vmt_shift_i * NOX_Rate_Truck) - _
    (rail_vmt_add_i * NOX_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
NOX_Tons_f = ((truck_vmt_shift_f * NOX_Rate_Truck) - _
```

```

(rail_vmt_add_f * NOX_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
PM10_Tons_i = ((truck_vmt_shift_i * PM10_Rate_Truck) - _
(rail_vmt_add_i * PM10_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
PM10_Tons_f = ((truck_vmt_shift_f * PM10_Rate_Truck) - _
(rail_vmt_add_f * PM10_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)

```

'Emission Sums

```

m_CO_Tons = (CO_Tons_f + CO_Tons_i) * Forecast_Period / 2
m_VOC_Tons = (VOC_Tons_f + VOC_Tons_i) * Forecast_Period / 2
m_NOX_Tons = (NOX_Tons_f + NOX_Tons_i) * Forecast_Period / 2
m_PM10_Tons = (PM10_Tons_f + PM10_Tons_i) * Forecast_Period / 2

```

'Emission Benefit Calculations

```

EnvBen_i = (CO_Tons_i * CO_Ton_Cost + VOC_Tons_i * VOCTon_Cost + _
NOX_Tons_i * NOXTon_Cost + PM10_Tons_i * PM10Ton_Cost) * (-1)
EnvBen_f = (CO_Tons_f * CO_Ton_Cost + VOC_Tons_f * VOCTon_Cost + _
NOX_Tons_f * NOXTon_Cost + PM10_Tons_f * PM10Ton_Cost) * (-1)
G_EnvBen = (EnvBen_f - EnvBen_i) / (Forecast_Period - 1)

```

'Emission NPV Calculations

```

m_Env_Ben = (EnvBen_i * PofA) + (G_EnvBen * PofG)

```

'Set module level values (to be displayed)

```

m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"
m_Values.Add m_PM10_Tons, "PM10_TONS"
m_Values.Add m_Env_Ben, "ENV_BEN"

```

'Safety Calculations

```

Fat_Rate_Rail = a_rst("Fat_Rate_Rail")
Inj_Rate_Rail = a_rst("Inj_Rate_Rail")
Prop_Rate_Rail = a_rst("Prop_Rate_Rail")
Evident_Cost = a_rst("Evident_Cost")
Fatality_Cost = a_rst("Fatality_Cost")
PDO_Cost = a_rst("PDO_Cost")

```

'Look up truck accident rates from lookup table

```

Fat_Rate_Truck = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_Truck)
If a_rst("Fat_Rate_Truck") <> Fat_Rate_Truck Or IsNull(a_rst("Fat_Rate_Truck")) Then
    Call UpdateRstField(a_rst, "Fat_Rate_Truck", Fat_Rate_Truck)
End If
Inj_Rate_Truck = LookupFatalityRate(m_dbs, 2, s_Injury, veht_Truck)
If a_rst("Inj_Rate_Truck") <> Inj_Rate_Truck Or IsNull(a_rst("Inj_Rate_Truck")) Then
    Call UpdateRstField(a_rst, "Inj_Rate_Truck", Inj_Rate_Truck)
End If
Prop_Rate_Truck = LookupFatalityRate(m_dbs, 2, s_Property, veht_Truck)
If a_rst("Prop_Rate_Truck") <> Prop_Rate_Truck Or IsNull(a_rst("Prop_Rate_Truck")) Then
    Call UpdateRstField(a_rst, "Prop_Rate_Truck", Prop_Rate_Truck)
End If

```

'Accident reduction due to shifted vmt

```

truck_Fatality_i = truck_vmt_shift_i * Fat_Rate_Truck / 1000000
truck_Fatality_f = truck_vmt_shift_f * Fat_Rate_Truck / 1000000
truck_Injury_i = truck_vmt_shift_i * Inj_Rate_Truck / 1000000
truck_Injury_f = truck_vmt_shift_f * Inj_Rate_Truck / 1000000

```

```
truck_Property_i = truck_vmt_shift_i * Prop_Rate_Truck / 1000000
truck_Property_f = truck_vmt_shift_f * Prop_Rate_Truck / 1000000
```

```
rail_Fatality_i = rail_vmt_add_i * Fat_Rate_Rail / 1000000
rail_Fatality_f = rail_vmt_add_f * Fat_Rate_Rail / 1000000
rail_Injury_i = rail_vmt_add_i * Inj_Rate_Rail / 1000000
rail_Injury_f = rail_vmt_add_f * Inj_Rate_Rail / 1000000
rail_Property_i = rail_vmt_add_i * Prop_Rate_Rail / 1000000
rail_Property_f = rail_vmt_add_f * Prop_Rate_Rail / 1000000
```

```
m_Fatality = ((truck_Fatality_i - rail_Fatality_i) + _
(truck_Fatality_f - rail_Fatality_f)) * (Forecast_Period) / 2
m_Injury = ((truck_Injury_i - rail_Injury_i) + _
(truck_Injury_f - rail_Injury_f)) * (Forecast_Period) / 2
m_Property = ((truck_Property_i - rail_Property_i) + _
(truck_Property_f - rail_Property_f)) * (Forecast_Period) / 2
```

'Calculate safety benefits

```
Safety_Ben_i = (((truck_Fatality_i - rail_Fatality) * Fatality_Cost) + _
((truck_Injury_i - rail_Injury_i) * Evident_Cost) + _
((truck_Property_i - rail_Property_i) * PDO_Cost)) * (-1)
Safety_Ben_f = (((truck_Fatality_f - rail_Fatality) * Fatality_Cost) + _
((truck_Injury_f - rail_Injury_f) * Evident_Cost) + _
((truck_Property_f - rail_Property_f) * PDO_Cost)) * (-1)
G_safety = (Safety_Ben_f - Safety_Ben_i) / (Forecast_Period - 1)
m_Safety_Ben = (Safety_Ben_i * PofA) + (G_safety * PofG)
```

'Set module level values (to be displayed)

```
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"
```

'Cost Calculations

'Capital Cost

For i = 1 To 5

```
m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
```

Next

```
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap
```

'Operations and Maintenance Cost

```
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
```



```

m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

```

```

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

```

```

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

```

```

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

```

```

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)

```

```

'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

```

```

'Environmental Retrofit Costs

```

```

For i = 1 To 5

```

```

m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))

```

```

Next

```

```

m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

```

```

'Environmental Retrofit Benefits

```

```

m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

```

```

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

```

```

'Benefit-Cost Calculations

```

```

Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben

```

```

If (m_Total_Cost - m_Terminal_Cost) = 0 Then
m_BCR = 0

```

```

Else

```

```

m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)

```

```

End If

```

```

If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then

```

```

m_WSDOT_BCR = 0

```

```

Else

```

```

m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)

```

```

End If

```

```

'Set module level values (to be displayed)

```

```

m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

```

'Calculation for the System Operation and Maintenance
m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
m_Sys_Pres = 100 * m_rst("Q2A")
m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
m_Sp_Needs = 100 * m_rst("Q3A")
m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
If m_rst("WTP_Corridor") Then
m_Cong_Rel = 50 + (m_rst("Q4") * 50)
Else
m_Cong_Rel = (m_rst("Q4") * 50)
End If
m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections

```

```

    m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
    m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
    If m_Safety_Ben > 0 Then
        m_Safety = 50 + (m_rst("Q7B") * 50)
    Else
        m_Safety = m_rst("Q7B") * 50
    End If
    m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
    'The -1 Multiplication Corrects the negative sign introduced for true
    m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E")))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") < 5 Then
        m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
    Else
        m_Collab = (m_rst("Q10A") * 50) + 50
    End If
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    If m_rst("Q14B") = 1 Then
        m_Air_Qual = 50 + m_rst("Q14A") * 50
    Else
        m_Air_Qual = 0
    End If
    m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

```

```
End Sub
```

```
Public Function GetItemValue(itemname As String) As Variant
```

```
    Dim rtnval
```

```
    rtnval = m_Values.Retrieve(itemname)
```

```
    GetItemValue = rtnval
```

```
End Function
```

```
Private Sub Class_Terminate()
```

```
    a_rst.Close
```

```
    m_rst.Close
```

```
    m_dbs.Close
```

```
    Set a_rst = Nothing
```

```
    Set m_rst = Nothing
```

```
    Set m_dbs = Nothing
```

```
    Set m_Values = Nothing
```

```
End Sub
```

Grade Separation / Crossing Improvement

Option Compare Database

'Variables for the class level database objects

```
Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection
```

'Variables used in intermediate calculations (variants)

```
Dim Forecast_Period
Dim m_Pers_TT_Sav
Dim m_Pers_TT_Ben
Dim m_Freight_TT_Sav
Dim m_Freight_TT_Ben
Dim m_TT_Sav
Dim m_WSDOT_Cap
Dim m_Federal_Cap
Dim m_Other1_Cap
Dim m_Other2_Cap
Dim m_Other3_Cap
Dim m_WSDOT_OM
Dim m_Federal_OM
Dim m_Other1_OM
Dim m_Other2_OM
Dim m_Other3_OM
Dim m_FishBarrier_Ben
Dim m_StormWater_Ben
Dim m_NoiseBarrier_Ben
Dim m_FishBarrier_Cap
Dim m_StormWater_Cap
Dim m_NoiseBarrier_Cap
```

'Variables to hold the calculated values (variants)

```
Dim m_Total_Benefit
Dim m_TT_Min
Dim m_TT_Ben
Dim m_User_Ben
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_EnvRetrofit_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
Dim m_Other2_Cost
Dim m_Other3_Cost
Dim m_Cap_Cost
```

```
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
Private Sub Class_Initialize()
    m_qryName = "rail_calc_Grade_Sep"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If
```

```

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")

'Calculate economic analysis factors
Discount_Rate = a_rst("Discount_Rate")
PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
    (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - _
    (Forecast_Period / ((1 + Discount_Rate) ^ Forecast_Period))
PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Travel Time Savings Calculations

'Passenger Travel Time Savings
ann_pass_PCI = m_rst("ann_pass_PCI")
ann_pass_PCF = m_rst("ann_pass_PCF")
Time_Value_Pers = a_rst("Time_Value_Pers")

If m_rst("avg_speed_pass_BC") <> 0 Then

```

```

    tt_pers_trip_bc = m_rst("proj_length") / m_rst("avg_speed_pass_BC")
End If
If m_rst("avg_speed_pass_PC") <> 0 Then
    tt_pers_trip_pc = m_rst("proj_length") / m_rst("avg_speed_pass_PC")
End If

```

```

tt_pers_sav_i = (tt_pers_trip_bc - tt_pers_trip_pc) * ann_pass_PCI
tt_pers_sav_f = (tt_pers_trip_bc - tt_pers_trip_pc) * ann_pass_PCF
m_Pers_TT_Sav = (tt_pers_sav_i + tt_pers_sav_f) * Forecast_Period / 2

```

```

tt_pers_ben_i = tt_pers_sav_i * Time_Value_Pers
tt_pers_ben_f = tt_pers_sav_f * Time_Value_Pers

```

```

G_tt_pers_ben = (tt_pers_ben_f - tt_pers_ben_i) / (Forecast_Period - 1)
m_Pers_TT_Ben = (tt_pers_ben_i * PofA) + (G_tt_pers_ben * PofG)

```

'Freight Travel Time Savings

```

Time_Value_Freight_Rail = a_rst("Time_Value_Freight_Rail")

```

```

tt_freight_trip_bc = m_rst("proj_length") / m_rst("avg_speed_freight_BC")
tt_freight_trip_pc = m_rst("proj_length") / m_rst("avg_speed_freight_PC")

```

```

tt_freight_sav_i = (tt_freight_trip_bc - tt_freight_trip_pc) * _
    m_rst("frt_trains_wk_i") * 52
tt_freight_sav_f = (tt_freight_trip_bc - tt_freight_trip_pc) * _
    m_rst("frt_trains_wk_f") * 52
m_Freight_TT_Sav = (tt_freight_sav_i + tt_freight_sav_f) * Forecast_Period / 2

```

```

tt_frt_ben_i = tt_freight_sav_i * Time_Value_Freight_Rail
tt_frt_ben_f = tt_freight_sav_f * Time_Value_Freight_Rail

```

```

G_tt_frt_ben = (tt_frt_ben_f - tt_frt_ben_i) / (Forecast_Period - 1)
m_Freight_TT_Ben = (tt_frt_ben_i * PofA) + (G_tt_frt_ben * PofG)

```

'Total Travel Time Calculations

```

m_TT_Sav = m_Pers_TT_Sav + m_Freight_TT_Sav
m_TT_Min = m_TT_Sav * 60
m_TT_Ben = m_Pers_TT_Ben + m_Freight_TT_Ben

```

'User Benefit and Environmental Calculations

```

'Since this procedure assumes no induced traffic, _
    no user or environmental benefits accrue
m_User_Ben = 0
m_Env_Ben = 0

```

'Set module level values (to be displayed)

```

m_Values.Add m_Forecast_Period, "FORECAST_PERIOD"
m_Values.Add m_Pers_TT_Sav, "PERS_TT_SAV"
m_Values.Add m_Pers_TT_Ben, "PERS_TT_BEN"
m_Values.Add m_Freight_TT_Sav, "FREIGHT_TT_SAV"
m_Values.Add m_Freight_TT_Ben, "FREIGHT_TT_BEN"
m_Values.Add m_TT_Sav, "TT_SAV"
m_Values.Add m_TT_Ben, "TT_BEN"
m_Values.Add m_User_Ben, "USER_BEN"

```



```

m_Values.Add m_Env_Ben, "ENV_BEN"

'Safety Calculations
Evident_Cost = a_rst("Evident_Cost")
Fatality_Cost = a_rst("Fatality_Cost")
PDO_Cost = a_rst("PDO_Cost")

'Accident reduction due to proposed project
acc_period = m_rst("end_acc_data") - m_rst("begin_acc_data")

If acc_period <> 0 Then
    ann_fatality = m_rst("prev_fatality") / acc_period
    ann_injury = m_rst("prev_injury") / acc_period
    ann_property = m_rst("prev_property") / acc_period
End If

m_Fatality = ann_fatality * Forecast_Period
m_Injury = ann_injury * Forecast_Period
m_Property = ann_property * Forecast_Period

'Calculate safety benefits
m_Safety_Ben = ((ann_fatality * Fatality_Cost) + (ann_injury * Evident_Cost) + _
(ann_property * PDO_Cost)) * PofA

'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
'Capital Cost
For i = 1 To 5
    m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

```

```

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
    m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
If (m_Total_Cost - m_Terminal_Cost) = 0 Then
    m_BCR = 0
Else
    m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
End If
If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
    m_WSDOT_BCR = 0
Else
    m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
End If
'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"

```

```

m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

```

'Calculation for the System Operation and Maintenance
m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
m_Sys_Pres = 100 * m_rst("Q2A")
m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
m_Sp_Needs = 100 * m_rst("Q3A")
m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
Else
    m_Cong_Rel = (m_rst("Q4") * 50)
End If
m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
If m_Safety_Ben > 0 Then

```

```

        m_Safety = 50 + (m_rst("Q7B") * 50)
    Else
        m_Safety = m_rst("Q7B") * 50
    End If
    m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
    'The -1 Multiplication Corrects the negative sign introduced for true
    m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E")))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") < 5 Then
        m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
    Else
        m_Collab = (m_rst("Q10A") * 50) + 50
    End If
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    If m_rst("Q14B") = 1 Then
        m_Air_Qual = 50 + m_rst("Q14A") * 50
    Else
        m_Air_Qual = 0
    End If
    m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

```

```
    GetItemValue = rtnval  
End Function  
  
Private Sub Class_Terminate()  
    a_rst.Close  
    m_rst.Close  
    m_dbs.Close  
    Set a_rst = Nothing  
    Set m_rst = Nothing  
    Set m_dbs = Nothing  
    Set m_Values = Nothing  
End Sub
```

Modal Connection Improvement

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim a_rst As DAO.Recordset

Dim m_qryName As String

Dim m_calcsComplete As Boolean

Dim m_Values As ValueCollection

'Variables used in intermediate calculations (variants)

Dim Forecast_Period

Dim m_TT_Sav

Dim m_OpCost

Dim m_Pavement_Sav

Dim m_Truck_Rem

Dim m_VMT_Shift

Dim m_Rail_Vmt_Add

Dim m_Approach

Dim m_WSDOT_Cap

Dim m_Federal_Cap

Dim m_Other1_Cap

Dim m_Other2_Cap

Dim m_Other3_Cap

Dim m_WSDOT_OM

Dim m_Federal_OM

Dim m_Other1_OM

Dim m_Other2_OM

Dim m_Other3_OM

Dim m_FishBarrier_Ben

Dim m_StormWater_Ben

Dim m_NoiseBarrier_Ben

Dim m_FishBarrier_Cap

Dim m_StormWater_Cap

Dim m_NoiseBarrier_Cap

'Variables to hold the calculated values (variants)

Dim m_Total_Benefit

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_EnvRetrofit_Ben

Dim m_Total_Cost

```

Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
Dim m_Other2_Cost
Dim m_Other3_Cost
Dim m_Cap_Cost
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR

```

'Variables to hold the outcome objective scores (variants)

```

Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource

```

'Speed Assumed for lookup table
Private Const TRUCKSPEED = 50

```

Private Sub Class_Initialize()
    m_qryName = "rail_calc_Modal_Connect"
    m_calcsComplete = False
End Sub

```

Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean

```

    Dim atype As String
    Dim qryDef As DAO.QueryDef

```

```

    Set m_dbs = CurrentDb

```

```

    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid

```

```

    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then

```

```

        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection

```

```

'If you are running MICA use the scenario assumptions
If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
    atype = "prj_Project_Assumptions"
Else
    atype = "prj_Global_Assumptions"
    pid = asmptnID
End If

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
    Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")

'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
        (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - _
        (Forecast_Period / ((1 + Discount_Rate) ^ Forecast_Period)))
    PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

```


'Time Savings Calculations

'Loading Time Savings

```

Time_Value_Freight_Rail = a_rst("Time_Value_Freight_Rail")
avg_loadspeed_BC = m_rst("avg_loadspeed_BC")
avg_loadspeed_PC = m_rst("avg_loadspeed_PC")

If (avg_loadspeed_PC - avg_loadspeed_BC) = 0 Then
  m_TT_Sav = 0
Else
  timesave_i = m_rst("brg_to_trn_BCI") / (avg_loadspeed_PC - avg_loadspeed_BC)
  timesave_f = m_rst("brg_to_trn_BCF") / (avg_loadspeed_PC - avg_loadspeed_BC)
  m_TT_Sav = (timesave_i + timesave_f) * Forecast_Period / 2
End If

```

```

m_TT_Min = m_TT_Sav * 60

```

```

timesave_ben_i = timesave_i * Time_Value_Freight_Rail
timesave_ben_f = timesave_f * Time_Value_Freight_Rail
G_timesave_ben = (timesave_ben_f - timesave_ben_i) / (Forecast_Period - 1)
m_TT_Ben = (timesave_ben_i * PofA) + (G_timesave_ben * PofG)

```

'Set module level values (to be displayed)

```

m_Values.Add Forecast_Period, "FORECAST_PERIOD"
m_Values.Add m_TT_Sav, "TT_SAV"
m_Values.Add m_TT_Min, "TT_MIN"
m_Values.Add m_TT_Ben, "TT_BEN"

```

'Shift from freight trucks to rail

```

general_tonpertruck = a_rst("general_tonpertruck")
freight_trip_length_BC = m_rst("freight_trip_length_BC")
freight_trip_length_PC = m_rst("freight_trip_length_PC")

truck_rem_i = (m_rst("brg_to_trn_BCI") - m_rst("brg_to_trn_PCI")) / _
  general_tonpertruck
truck_rem_f = (m_rst("brg_to_trn_BCF") - m_rst("brg_to_trn_PCF")) / _
  general_tonpertruck

m_Truck_Rem = (truck_rem_i + truck_rem_f) * Forecast_Period / 2

```

'Truck VMT shift

```

truck_vmt_shift_i = truck_rem_i * freight_trip_length_BC
truck_vmt_shift_f = truck_rem_f * freight_trip_length_BC
m_VMT_Shift = (truck_vmt_shift_i + truck_vmt_shift_f) * Forecast_Period / 2

```

'Added Rail VMT

```

rail_vmt_add_i = ((m_rst("freight_train_wk_PCI") * freight_trip_length_PC) - _
  (m_rst("freight_train_wk_BCI") * freight_trip_length_BC)) * 52
rail_vmt_add_f = ((m_rst("freight_train_wk_PCF") * freight_trip_length_PC) - _
  (m_rst("freight_train_wk_BCF") * freight_trip_length_BC)) * 52
m_Rail_Vmt_Add = (rail_vmt_add_i + rail_vmt_add_f) * Forecast_Period / 2

```

'Set module level values (to be displayed)

```

m_Values.Add m_Truck_Rem, "TRUCK_REM"

```

```

m_Values.Add m_VMT_Shift, "VMT_SHIFT"
m_Values.Add m_Rail_Vmt_Add, "RAIL_VMT_ADD"

```

'Operating Cost Savings Calculations

```

Truck_OpCost_Full = a_rst("Truck_OpCost_Full")
Truck_OpCost_Direct = a_rst("Truck_OpCost_Direct")
If a_rst("Full_Cost") Then
  m_Approach = "Full cost"
  OpCost_i = (-1) * truck_vmt_shift_i * Truck_OpCost_Full
  OpCost_f = (-1) * truck_vmt_shift_f * Truck_OpCost_Full
Else
  m_Approach = "Direct cost"
  OpCost_i = (-1) * truck_vmt_shift_i * Truck_OpCost_Direct
  OpCost_f = (-1) * truck_vmt_shift_f * Truck_OpCost_Direct
End If

G_Opcost = (OpCost_f - OpCost_i) / (Forecast_Period - 1)
m_OpCost = (OpCost_i * PofA) + (G_Opcost * PofG)

```

'Pavement Cost Savings

```

truck_pave_cost = a_rst("truck_pave_cost")
Pavement_Sav_i = (-1) * truck_vmt_shift_i * truck_pave_cost
Pavement_Sav_f = (-1) * truck_vmt_shift_f * truck_pave_cost

G_Pavement_Sav = (Pavement_Sav_f - Pavement_Sav_i) / (Forecast_Period - 1)
m_Pavement_Sav = (Pavement_Sav_i * PofA) + (G_Pavement_Sav * PofG)

```

'User Benefit NPV Calculation

```

m_User_Ben = m_OpCost + m_Pavement_Sav

```

'Set module level values (to be displayed)

```

m_Values.Add m_Approach, "APPROACH"
m_Values.Add m_OpCost, "OPCOST"
m_Values.Add m_Pavement_Sav, "PAVEMENT_SAV"
m_Values.Add m_User_Ben, "USER_BEN"

```

'Air Pollution Calculations - for diverted auto trips

```

CO_Rate_Truck = LookupAirPollution(m_dbs, TRUCKSPEED, em_CO, veht_Truck)

If a_rst("CO_Rate_Truck") <> CO_Rate_Truck Or IsNull(a_rst("CO_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "CO_Rate_Truck", CO_Rate_Truck)
End If
CO_Rate_Rail = a_rst("CO_Rate_Rail")
CO_Ton_Cost = a_rst("CO_Ton_Cost")
VOC_Rate_Truck = a_rst("VOC_Rate_Truck")
VOC_Rate_Rail = a_rst("VOC_Rate_Rail")
VOCTon_Cost = a_rst("VOCTon_Cost")
NOX_Rate_Truck = LookupAirPollution(m_dbs, TRUCKSPEED, em_NOX, veht_Truck)
If a_rst("NOX_Rate_Truck") <> NOX_Rate_Truck Or IsNull(a_rst("NOX_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "NOX_Rate_Truck", NOX_Rate_Truck)
End If
NOX_Rate_Rail = a_rst("NOX_Rate_Rail")
NOXTon_Cost = a_rst("NOXTon_Cost")
PM10_Rate_Truck = a_rst("PM10_Rate_Truck")
PM10_Rate_Rail = a_rst("PM10_Rate_Rail")

```

```

PM10Ton_Cost = a_rst("PM10Ton_Cost")

CO_Tons_i = ((truck_vmt_shift_i * CO_Rate_Truck) - _
  (rail_vmt_add_i * CO_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
CO_Tons_f = ((truck_vmt_shift_f * CO_Rate_Truck) - _
  (rail_vmt_add_f * CO_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
VOC_Tons_i = ((truck_vmt_shift_i * VOC_Rate_Truck) - _
  (rail_vmt_add_i * VOC_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
VOC_Tons_f = ((truck_vmt_shift_f * VOC_Rate_Truck) - _
  (rail_vmt_add_f * VOC_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
NOX_Tons_i = ((truck_vmt_shift_i * NOX_Rate_Truck) - _
  (rail_vmt_add_i * NOX_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
NOX_Tons_f = ((truck_vmt_shift_f * NOX_Rate_Truck) - _
  (rail_vmt_add_f * NOX_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
PM10_Tons_i = ((truck_vmt_shift_i * PM10_Rate_Truck) - _
  (rail_vmt_add_i * PM10_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
PM10_Tons_f = ((truck_vmt_shift_f * PM10_Rate_Truck) - _
  (rail_vmt_add_f * PM10_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)

'Emission Sums
  m_CO_Tons = (CO_Tons_f + CO_Tons_i) * Forecast_Period / 2
  m_VOC_Tons = (VOC_Tons_f + VOC_Tons_i) * Forecast_Period / 2
  m_NOX_Tons = (NOX_Tons_f + NOX_Tons_i) * Forecast_Period / 2
  m_PM10_Tons = (PM10_Tons_f + PM10_Tons_i) * Forecast_Period / 2

'Emission Benefit Calculations
EnvBen_i = (CO_Tons_i * COTon_Cost + VOC_Tons_i * VOCTon_Cost + _
  NOX_Tons_i * NOXTon_Cost + PM10_Tons_i * PM10Ton_Cost) * (-1)
EnvBen_f = (CO_Tons_f * COTon_Cost + VOC_Tons_f * VOCTon_Cost + _
  NOX_Tons_f * NOXTon_Cost + PM10_Tons_f * PM10Ton_Cost) * (-1)
G_EnvBen = (EnvBen_f - EnvBen_i) / (Forecast_Period - 1)

'Emission NPV Calculations
m_Env_Ben = (EnvBen_i * PofA) + (G_EnvBen * PofG)

'Set module level values (to be displayed)
m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"
m_Values.Add m_PM10_Tons, "PM10_TONS"
m_Values.Add m_Env_Ben, "ENV_BEN"

'Safety Calculations
Fat_Rate_Rail = a_rst("Fat_Rate_Rail")
Inj_Rate_Rail = a_rst("Inj_Rate_Rail")
Prop_Rate_Rail = a_rst("Prop_Rate_Rail")
Evident_Cost = a_rst("Evident_Cost")
Fatality_Cost = a_rst("Fatality_Cost")
PDO_Cost = a_rst("PDO_Cost")

'Look up truck accident rates from lookup table
Fat_Rate_Truck = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_Truck)
If a_rst("Fat_Rate_Truck") <> Fat_Rate_Truck Or IsNull(a_rst("Fat_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Fat_Rate_Truck", Fat_Rate_Truck)
End If

```

```

Inj_Rate_Truck = LookupFatalityRate(m_dbs, 2, s_Injury, veht_Truck)
If a_rst("Inj_Rate_Truck") <> Inj_Rate_Truck Or IsNull(a_rst("Inj_Rate_Truck")) Then
    Call UpdateRstField(a_rst, "Inj_Rate_Truck", Inj_Rate_Truck)
End If
Prop_Rate_Truck = LookupFatalityRate(m_dbs, 2, s_Property, veht_Truck)
If a_rst("Prop_Rate_Truck") <> Prop_Rate_Truck Or IsNull(a_rst("Prop_Rate_Truck")) Then
    Call UpdateRstField(a_rst, "Prop_Rate_Truck", Prop_Rate_Truck)
End If

```

'Accident reduction due to shifted vmt

```

truck_Fatality_i = truck_vmt_shift_i * Fat_Rate_Truck / 1000000
truck_Fatality_f = truck_vmt_shift_f * Fat_Rate_Truck / 1000000
truck_Injury_i = truck_vmt_shift_i * Inj_Rate_Truck / 1000000
truck_Injury_f = truck_vmt_shift_f * Inj_Rate_Truck / 1000000
truck_Property_i = truck_vmt_shift_i * Prop_Rate_Truck / 1000000
truck_Property_f = truck_vmt_shift_f * Prop_Rate_Truck / 1000000

```

```

rail_Fatality_i = rail_vmt_add_i * Fat_Rate_Rail / 1000000
rail_Fatality_f = rail_vmt_add_f * Fat_Rate_Rail / 1000000
rail_Injury_i = rail_vmt_add_i * Inj_Rate_Rail / 1000000
rail_Injury_f = rail_vmt_add_f * Inj_Rate_Rail / 1000000
rail_Property_i = rail_vmt_add_i * Prop_Rate_Rail / 1000000
rail_Property_f = rail_vmt_add_f * Prop_Rate_Rail / 1000000

```

```

m_Fatality = ((truck_Fatality_i - rail_Fatality_i) + _
    (truck_Fatality_f - rail_Fatality_f)) * (Forecast_Period) / 2
m_Injury = ((truck_Injury_i - rail_Injury_i) + _
    (truck_Injury_f - rail_Injury_f)) * (Forecast_Period) / 2
m_Property = ((truck_Property_i - rail_Property_i) + _
    (truck_Property_f - rail_Property_f)) * (Forecast_Period) / 2

```

'Calculate safety benefits

```

Safety_Ben_i = (((truck_Fatality_i - rail_Fatality_i) * Fatality_Cost) + _
    ((truck_Injury_i - rail_Injury_i) * Evident_Cost) + _
    ((truck_Property_i - rail_Property_i) * PDO_Cost)) * (-1)
Safety_Ben_f = (((truck_Fatality_f - rail_Fatality_f) * Fatality_Cost) + _
    ((truck_Injury_f - rail_Injury_f) * Evident_Cost) + _
    ((truck_Property_f - rail_Property_f) * PDO_Cost)) * (-1)
G_safety = (Safety_Ben_f - Safety_Ben_i) / (Forecast_Period - 1)
m_Safety_Ben = (Safety_Ben_i * PofA) + (G_safety * PofG)

```

'Set module level values (to be displayed)

```

m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

```

'Cost Calculations

'Capital Cost

For i = 1 To 5

```

m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))

```

```

    m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
      1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
      1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  Next
  m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
  m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

  Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
  m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations

```

```

    Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
    If (m_Total_Cost - m_Terminal_Cost) = 0 Then
        m_BCR = 0
    Else
        m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
    End If
    If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
        m_WSDOT_BCR = 0
    Else
        m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
    End If
    'Set module level values (to be displayed)
    m_Values.Add m_Total_Cost, "TOTAL_COST"
    m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
    m_Values.Add m_Federal_Cost, "FEDERAL_COST"
    m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
    m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
    m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
    m_Values.Add m_Other1_Cost, "OTHER1_COST"
    m_Values.Add m_Other2_Cost, "OTHER2_COST"
    m_Values.Add m_Other3_Cost, "OTHER3_COST"
    m_Values.Add m_Cap_Cost, "CAP_COST"
    m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
    m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
    m_Values.Add m_Other1_Cap, "OTHER1_CAP"
    m_Values.Add m_Other2_Cap, "OTHER2_CAP"
    m_Values.Add m_Other3_Cap, "OTHER3_CAP"
    m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
    m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
    m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
    m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
    m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
    m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
    m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
    m_Values.Add m_WSDOT_OM, "WSDOT_OM"
    m_Values.Add m_Federal_OM, "FEDERAL_OM"
    m_Values.Add m_Other1_OM, "OTHER1_OM"
    m_Values.Add m_Other2_OM, "OTHER2_OM"
    m_Values.Add m_Other3_OM, "OTHER3_OM"
    m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
    m_Values.Add m_BCR, "BCR"
    m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next
    'Calculation for the System Operation and Maintenance
    m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
    m_Values.Add m_Sys_OM, "SYS_OM"
    'Calculation for the System Preservation
    m_Sys_Pres = 100 * m_rst("Q2A")
    m_Values.Add m_Sys_Pres, "SYS_PRES"
    'Calculation for the Special Needs Transportation

```

```

    m_Sp_Needs = 100 * m_rst("Q3A")
    m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
    If m_rst("WTP_Corridor") Then
        m_Cong_Rel = 50 + (m_rst("Q4") * 50)
    Else
        m_Cong_Rel = (m_rst("Q4") * 50)
    End If
    m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
    m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
    m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
    m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
    m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
    If m_Safety_Ben > 0 Then
        m_Safety = 50 + (m_rst("Q7B") * 50)
    Else
        m_Safety = m_rst("Q7B") * 50
    End If
    m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
    'The -1 Multiplication Corrects the negative sign introduced for true
    m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E")))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") < 5 Then
        m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
    Else
        m_Collab = (m_rst("Q10A") * 50) + 50
    End If
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    If m_rst("Q14B") = 1 Then
        m_Air_Qual = 50 + m_rst("Q14A") * 50
    Else
        m_Air_Qual = 0
    End If
    m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _

```

```
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub
```


Passenger Trainset Purchase

Option Compare Database

'Variables for the class level database objects

```
Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection
```

'Variables used in intermediate calculations (variants)

```
Dim Forecast_Period
Dim m_OpCost
Dim m_Fare_Cost
Dim m_VMT_Shift
Dim m_Rail_Vmt_Add
Dim m_Approach
Dim m_WSDOT_Cap
Dim m_Federal_Cap
Dim m_Other1_Cap
Dim m_Other2_Cap
Dim m_Other3_Cap
Dim m_WSDOT_OM
Dim m_Federal_OM
Dim m_Other1_OM
Dim m_Other2_OM
Dim m_Other3_OM
```

'Variables to hold the calculated values (variants)

```
Dim m_Total_Benefit
Dim m_TT_Ben
Dim m_User_Ben
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
Dim m_Other2_Cost
Dim m_Other3_Cost
Dim m_Cap_Cost
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

```

'Variables to hold the outcome objective scores (variants)
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource

'Speed Assumed for lookup table
Private Const AUTOSPEED = 50

Private Sub Class_Initialize()
    m_qryName = "rail_calc_Trainset"
    m_calcsComplete = False
End Sub

Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean

    Dim atype As String
    Dim qryDef As DAO.QueryDef

    Set m_dbs = CurrentDb

    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid

    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then

        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection

        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If

        'Open up a assumption recordset - depend on if in MICA or not

```

```

Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
    'Calculate Forecast Period
    Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")

'Calculate economic analysis factors
Discount_Rate = a_rst("Discount_Rate")
PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
    (Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / _
    ((1 + Discount_Rate) ^ Forecast_Period))
PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Person shift
pers_shift_i = m_rst("ann_pass_PCI") - m_rst("ann_pass_BCI")
pers_shift_f = m_rst("ann_pass_PCF") - m_rst("ann_pass_BCF")
m_Pers_Shift = (pers_shift_i + pers_shift_f) * Forecast_Period / 2

'Auto VMT shift
avo = a_rst("avo")
pass_trip_length_PC = m_rst("pass_trip_length_PC")
auto_divert_rail = m_rst("auto_divert_rail")

```

```

auto_vmt_shift_i = (-1) * pers_shift_i * auto_divert_rail * _
    pass_trip_length_PC / avo
auto_vmt_shift_f = (-1) * pers_shift_f * auto_divert_rail * _
    pass_trip_length_PC / avo
m_VMT_Shift = (auto_vmt_shift_i + auto_vmt_shift_f) * Forecast_Period / 2

```

```

'Added Rail VMT
pass_trip_length_BC = m_rst("pass_trip_length_BC")

```

```

rail_vmt_add_i = ((m_rst("pass_train_wk_PCI") * pass_trip_length_PC) - _
    (m_rst("pass_train_wk_BCI") * pass_trip_length_BC)) * 52
rail_vmt_add_f = ((m_rst("pass_train_wk_PCF") * pass_trip_length_PC) - _
    (m_rst("pass_train_wk_BCF") * pass_trip_length_BC)) * 52
m_Rail_Vmt_Add = (rail_vmt_add_i + rail_vmt_add_f) * Forecast_Period / 2

```

```

'Set module level values (to be displayed)
m_Values.Add Forecast_Period, "FORECAST_PERIOD"
m_Values.Add m_Pers_Shift, "PERS_SHIFT"
m_Values.Add m_VMT_Shift, "VMT_SHIFT"
m_Values.Add m_Rail_Vmt_Add, "RAIL_VMT_ADD"

```

'Travel Time Savings Calculations

```

'No travel time benefits are calculated for this project type
m_TT_Ben = 0

```

'Operating Cost Savings Calculations

```

Veh_OpCost_Full = a_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = a_rst("Veh_OpCost_Direct")
If a_rst("Full_Cost") Then
    m_Approach = "Full cost"
    OpCost_i = (-1) * auto_vmt_shift_i * Veh_OpCost_Full
    OpCost_f = (-1) * auto_vmt_shift_f * Veh_OpCost_Full
Else
    m_Approach = "Direct cost"
    OpCost_i = (-1) * auto_vmt_shift_i * Veh_OpCost_Direct
    OpCost_f = (-1) * auto_vmt_shift_f * Veh_OpCost_Direct
End If

```

```

G_Opcost = (OpCost_f - OpCost_i) / (Forecast_Period - 1)
m_OpCost = (OpCost_i * PofA) + (G_Opcost * PofG)

```

'Out-of-pocket costs to users due to fares

```

Fare = m_rst("fare")
fare_cost_i = pers_shift_i * Fare
fare_cost_f = pers_shift_f * Fare
G_fare_cost = (fare_cost_f - fare_cost_i) / (Forecast_Period - 1)
m_Fare_Cost = -((fare_cost_i * PofA) + (G_fare_cost * PofG))

```

'User Benefit NPV Calculation

```

m_User_Ben = m_OpCost + m_Fare_Cost

```

'Set module level values (to be displayed)

```

m_Values.Add m_Approach, "APPROACH"
m_Values.Add m_OpCost, "OPCOST"
m_Values.Add m_Fare_Cost, "FARE_COST"

```

```

m_Values.Add m_User_Ben, "USER_BEN"
m_Values.Add m_TT_Ben, "TT_BEN"

```

```
'Air Pollution Calculations - for diverted auto trips
```

```
CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)
```

```
If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
```

```
    Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
```

```
End If
```

```
CO_Rate_Rail = a_rst("CO_Rate_Rail")
```

```
COTon_Cost = a_rst("COTon_Cost")
```

```
VOC_Rate_Auto = a_rst("VOC_Rate_Auto")
```

```
VOC_Rate_Rail = a_rst("VOC_Rate_Rail")
```

```
VOCTon_Cost = a_rst("VOCTon_Cost")
```

```
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
```

```
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
```

```
    Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
```

```
End If
```

```
NOX_Rate_Rail = a_rst("NOX_Rate_Rail")
```

```
NOXTon_Cost = a_rst("NOXTon_Cost")
```

```
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")
```

```
PM10_Rate_Rail = a_rst("PM10_Rate_Rail")
```

```
PM10Ton_Cost = a_rst("PM10Ton_Cost")
```

```
CO_Tons_i = ((auto_vmt_shift_i * CO_Rate_Auto) - _
    (rail_vmt_add_i * CO_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
```

```
CO_Tons_f = ((auto_vmt_shift_f * CO_Rate_Auto) - _
    (rail_vmt_add_f * CO_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
```

```
VOC_Tons_i = ((auto_vmt_shift_i * VOC_Rate_Auto) - _
    (rail_vmt_add_i * VOC_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
```

```
VOC_Tons_f = ((auto_vmt_shift_f * VOC_Rate_Auto) - _
    (rail_vmt_add_f * VOC_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
```

```
NOX_Tons_i = ((auto_vmt_shift_i * NOX_Rate_Auto) - _
    (rail_vmt_add_i * NOX_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
```

```
NOX_Tons_f = ((auto_vmt_shift_f * NOX_Rate_Auto) - _
    (rail_vmt_add_f * NOX_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
```

```
PM10_Tons_i = ((auto_vmt_shift_i * PM10_Rate_Auto) - _
    (rail_vmt_add_i * PM10_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
```

```
PM10_Tons_f = ((auto_vmt_shift_f * PM10_Rate_Auto) - _
    (rail_vmt_add_f * PM10_Rate_Rail)) * (1 / 1000) * (0.9842 / 1000)
```

```
'Emission Sums
```

```
m_CO_Tons = (CO_Tons_f + CO_Tons_i) * Forecast_Period / 2
```

```
m_VOC_Tons = (VOC_Tons_f + VOC_Tons_i) * Forecast_Period / 2
```

```
m_NOX_Tons = (NOX_Tons_f + NOX_Tons_i) * Forecast_Period / 2
```

```
m_PM10_Tons = (PM10_Tons_f + PM10_Tons_i) * Forecast_Period / 2
```

```
'Emission Benefit Calculations
```

```
EnvBen_i = (CO_Tons_i * COTon_Cost + VOC_Tons_i * VOCTon_Cost + _
    NOX_Tons_i * NOXTon_Cost + PM10_Tons_i * PM10Ton_Cost) * (-1)
```

```
EnvBen_f = (CO_Tons_f * COTon_Cost + VOC_Tons_f * VOCTon_Cost + _
    NOX_Tons_f * NOXTon_Cost + PM10_Tons_f * PM10Ton_Cost) * (-1)
```

```
G_EnvBen = (EnvBen_f - EnvBen_i) / (Forecast_Period - 1)
```

```
'Emission NPV Calculations
```

```
m_Env_Ben = (EnvBen_i * PofA) + (G_EnvBen * PofG)
```

```
'Set module level values (to be displayed)
  m_Values.Add m_CO_Tons, "CO_TONS"
  m_Values.Add m_VOC_Tons, "VOC_TONS"
  m_Values.Add m_NOX_Tons, "NOX_TONS"
  m_Values.Add m_PM10_Tons, "PM10_TONS"
  m_Values.Add m_Env_Ben, "ENV_BEN"
```

```
'Safety Calculations
```

```
Fat_Rate_Rail = a_rst("Fat_Rate_Rail")
Inj_Rate_Rail = a_rst("Inj_Rate_Rail")
Prop_Rate_Rail = a_rst("Prop_Rate_Rail")
Evident_Cost = a_rst("Evident_Cost")
Fatality_Cost = a_rst("Fatality_Cost")
PDO_Cost = a_rst("PDO_Cost")
```

```
'Look up accident rates from lookup table
```

```
Fat_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_auto)
If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
End If
Inj_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Injury, veht_auto)
If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
End If
Prop_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Property, veht_auto)
If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
End If
```

```
'Accident reduction due to shifted vmt
```

```
auto_Fatality_i = auto_vmt_shift_i * Fat_Rate_Auto / 1000000
auto_Fatality_f = auto_vmt_shift_f * Fat_Rate_Auto / 1000000
auto_Injury_i = auto_vmt_shift_i * Inj_Rate_Auto / 1000000
auto_Injury_f = auto_vmt_shift_f * Inj_Rate_Auto / 1000000
auto_Property_i = auto_vmt_shift_i * Prop_Rate_Auto / 1000000
auto_Property_f = auto_vmt_shift_f * Prop_Rate_Auto / 1000000
```

```
rail_Fatality_i = rail_vmt_add_i * Fat_Rate_Rail / 1000000
rail_Fatality_f = rail_vmt_add_f * Fat_Rate_Rail / 1000000
rail_Injury_i = rail_vmt_add_i * Inj_Rate_Rail / 1000000
rail_Injury_f = rail_vmt_add_f * Inj_Rate_Rail / 1000000
rail_Property_i = rail_vmt_add_i * Prop_Rate_Rail / 1000000
rail_Property_f = rail_vmt_add_f * Prop_Rate_Rail / 1000000
```

```
m_Fatality = ((rail_Fatality_i + auto_Fatality_i) + _
  (rail_Fatality_f + auto_Fatality_f)) * (Forecast_Period) / 2
m_Injury = ((rail_Injury_i + auto_Injury_i) + _
  (rail_Injury_f + auto_Injury_f)) * (Forecast_Period) / 2
m_Property = ((rail_Property_i + auto_Property_i) + _
  (rail_Property_f + auto_Property_f)) * (Forecast_Period) / 2
```

```
'Calculate safety benefits
```

```
Safety_Ben_i = (((auto_Fatality_i + rail_Fatality) * Fatality_Cost) + _
  ((auto_Injury_i + rail_Injury_i) * Evident_Cost) + _
```

```

    ((auto_Property_i + rail_Property_i) * PDO_Cost)) * (-1)
    Safety_Ben_f = (((auto_Fatality_f + rail_Fatality_f) * Fatality_Cost) + _
    ((auto_Injury_f + rail_Injury_f) * Evident_Cost) + _
    ((auto_Property_f + rail_Property_f) * PDO_Cost)) * (-1)
    G_safety = (Safety_Ben_f - Safety_Ben_i) / (Forecast_Period - 1)
    m_Safety_Ben = (Safety_Ben_i * PofA) + (G_safety * PofG)

```

```

'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

```

'Cost Calculations

'Capital Cost

For i = 1 To 5

```

    m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))

```

Next

```

m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

```

'Operations and Maintenance Cost

```

m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

```

'Terminal cost

```

m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

```

'Total Costs

```

m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

```

```

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

```

```

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)

```

'Environmental Retrofit Calculations

```

fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")

```

```

noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
  m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations
Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
If (m_Total_Cost - m_Terminal_Cost) = 0 Then
  m_BCR = 0
Else
  m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
End If
If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
  m_WSDOT_BCR = 0
Else
  m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
End If
'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"

```



```

m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

```

'Calculation for the System Operation and Maintenance
m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
m_Sys_Pres = 100 * m_rst("Q2A")
m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
m_Sp_Needs = 100 * m_rst("Q3A")
m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
Else
    m_Cong_Rel = (m_rst("Q4") * 50)
End If
m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
Else
    m_Safety = m_rst("Q7B") * 50
End If
m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
m_Security = 100 * m_rst("Q8A")
m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
'The -1 Multiplication Corrects the negative sign introduced for true
m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)

```

```

Else
    m_Collab = (m_rst("Q10A") * 50) + 50
End If
m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + m_rst("Q14A") * 50
Else
    m_Air_Qual = 0
End If
m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
(33 * m_rst("Q15C"))
m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
(25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
(1 + m_rst("Q16E") + m_rst("Q16F"))
m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
m_Resource = 100 * m_rst("Q17")
m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub

```

Station Improvement

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim a_rst As DAO.Recordset

Dim m_qryName As String

Dim m_calcsComplete As Boolean

Dim m_Values As ValueCollection

'Variables used in intermediate calculations (variants)

Dim m_Forecast_Period

Dim m_OpCost

Dim m_Fare_Cost

Dim m_VMT_Shift

Dim m_Approach

Dim m_WSDOT_Cap

Dim m_Federal_Cap

Dim m_Other1_Cap

Dim m_Other2_Cap

Dim m_Other3_Cap

Dim m_WSDOT_OM

Dim m_Federal_OM

Dim m_Other1_OM

Dim m_Other2_OM

Dim m_Other3_OM

Dim m_FishBarrier_Ben

Dim m_StormWater_Ben

Dim m_NoiseBarrier_Ben

Dim m_FishBarrier_Cap

Dim m_StormWater_Cap

Dim m_NoiseBarrier_Cap

'Variables to hold the calculated values (variants)

Dim m_Total_Benefit

Dim m_TT_Ben

Dim m_User_Ben

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_EnvRetrofit_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Other1_Cost

Dim m_Other2_Cost

```
Dim m_Other3_Cost
Dim m_Cap_Cost
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
'Speed Assumed for lookup table
Private Const AUTOSPEED = 50
```

```
Private Sub Class_Initialize()
    m_qryName = "rail_calc_Station"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
```

```

Else
    atype = "prj_Global_Assumptions"
    pid = asmptnID
End If

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")

'Calculate economic analysis factors
Discount_Rate = a_rst("Discount_Rate")
PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
(Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / _
((1 + Discount_Rate) ^ Forecast_Period))
PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Person shift
pers_shift_i = m_rst("ann_pass_PCI") - m_rst("ann_pass_BCI")
pers_shift_f = m_rst("ann_pass_PCF") - m_rst("ann_pass_BCF")

```

```

m_Pers_Shift = (pers_shift_i + pers_shift_f) * Forecast_Period / 2

'Auto VMT shift
avo = a_rst("avo")
pass_trip_length_PC = m_rst("pass_trip_length_PC")
auto_divert_rail = m_rst("auto_divert_rail")

auto_vmt_shift_i = (-1) * pers_shift_i * auto_divert_rail * _
    pass_trip_length_PC / avo
auto_vmt_shift_f = (-1) * pers_shift_f * auto_divert_rail * _
    pass_trip_length_PC / avo
m_VMT_Shift = (auto_vmt_shift_i + auto_vmt_shift_f) * Forecast_Period / 2

'Set module level values (to be displayed)
m_Values.Add m_Forecast_Period, "FORECAST_PERIOD"
m_Values.Add m_Pers_Shift, "PERS_SHIFT"
m_Values.Add m_VMT_Shift, "VMT_SHIFT"

'Operating Cost Savings Calculations
Veh_OpCost_Full = a_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = a_rst("Veh_OpCost_Direct")
If a_rst("Full_Cost") Then
    m_Approach = "Full cost"
    OpCost_i = (-1) * auto_vmt_shift_i * Veh_OpCost_Full
    OpCost_f = (-1) * auto_vmt_shift_f * Veh_OpCost_Full
Else
    m_Approach = "Direct cost"
    OpCost_i = (-1) * auto_vmt_shift_i * Veh_OpCost_Direct
    OpCost_f = (-1) * auto_vmt_shift_f * Veh_OpCost_Direct
End If

G_Opcost = (OpCost_f - OpCost_i) / (Forecast_Period - 1)
m_OpCost = (OpCost_i * PofA) + (G_Opcost * PofG)

'Out-of-pocket costs to users due to fares
Fare = m_rst("fare")
fare_cost_i = pers_shift_i * Fare
fare_cost_f = pers_shift_f * Fare
G_fare_cost = (fare_cost_f - fare_cost_i) / (Forecast_Period - 1)
m_Fare_Cost = -(fare_cost_i * PofA) + (G_fare_cost * PofG)

'User Benefit NPV Calculation
m_User_Ben = m_OpCost + m_Fare_Cost

'User Benefit NPV Calculation
m_User_Ben = m_OpCost + m_Fare_Cost

'Travel Time Savings Calculations
'No travel time benefits are calculated for this project type
m_TT_Ben = 0

'Set module level values (to be displayed)
m_Values.Add m_Approach, "APPROACH"
m_Values.Add m_OpCost, "OPCOST"

```

```

m_Values.Add m_Fare_Cost, "FARE_COST"
m_Values.Add m_User_Ben, "USER_BEN"
m_Values.Add m_TT_Ben, "TT_BEN"

```

```
'Air Pollution Calculations - for diverted auto trips
```

```
CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)
```

```
If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
```

```
    Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
```

```
End If
```

```
CO_Rate_Rail = a_rst("CO_Rate_Rail")
```

```
COTon_Cost = a_rst("COTon_Cost")
```

```
VOC_Rate_Auto = a_rst("VOC_Rate_Auto")
```

```
VOC_Rate_Rail = a_rst("VOC_Rate_Rail")
```

```
VOCTon_Cost = a_rst("VOCTon_Cost")
```

```
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
```

```
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
```

```
    Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
```

```
End If
```

```
NOX_Rate_Rail = a_rst("NOX_Rate_Rail")
```

```
NOXTon_Cost = a_rst("NOXTon_Cost")
```

```
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")
```

```
PM10_Rate_Rail = a_rst("PM10_Rate_Rail")
```

```
PM10Ton_Cost = a_rst("PM10Ton_Cost")
```

```
CO_Tons_i = (auto_vmt_shift_i * CO_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
```

```
CO_Tons_f = (auto_vmt_shift_f * CO_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
```

```
VOC_Tons_i = (auto_vmt_shift_i * VOC_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
```

```
VOC_Tons_f = (auto_vmt_shift_f * VOC_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
```

```
NOX_Tons_i = (auto_vmt_shift_i * NOX_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
```

```
NOX_Tons_f = (auto_vmt_shift_f * NOX_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
```

```
PM10_Tons_i = (auto_vmt_shift_i * PM10_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
```

```
PM10_Tons_f = (auto_vmt_shift_f * PM10_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
```

```
'Emission Sums
```

```
    m_CO_Tons = (CO_Tons_f + CO_Tons_i) * Forecast_Period / 2
```

```
    m_VOC_Tons = (VOC_Tons_f + VOC_Tons_i) * Forecast_Period / 2
```

```
    m_NOX_Tons = (NOX_Tons_f + NOX_Tons_i) * Forecast_Period / 2
```

```
    m_PM10_Tons = (PM10_Tons_f + PM10_Tons_i) * Forecast_Period / 2
```

```
'Emission Benefit Calculations
```

```
EnvBen_i = (CO_Tons_i * COTon_Cost + VOC_Tons_i * VOCTon_Cost + _
```

```
    NOX_Tons_i * NOXTon_Cost + PM10_Tons_i * PM10Ton_Cost) * (-1)
```

```
EnvBen_f = (CO_Tons_f * COTon_Cost + VOC_Tons_f * VOCTon_Cost + _
```

```
    NOX_Tons_f * NOXTon_Cost + PM10_Tons_f * PM10Ton_Cost) * (-1)
```

```
G_EnvBen = (EnvBen_f - EnvBen_i) / (Forecast_Period - 1)
```

```
'Emission NPV Calculations
```

```
m_Env_Ben = (EnvBen_i * PofA) + (G_EnvBen * PofG)
```

```
'Set module level values (to be displayed)
```

```
m_Values.Add m_CO_Tons, "CO_TONS"
```

```
m_Values.Add m_VOC_Tons, "VOC_TONS"
```

```
m_Values.Add m_NOX_Tons, "NOX_TONS"
```

```
m_Values.Add m_PM10_Tons, "PM10_TONS"
```

```
m_Values.Add m_Env_Ben, "ENV_BEN"
```

```

'Safety Calculations
  Evident_Cost = a_rst("Evident_Cost")
  Fatality_Cost = a_rst("Fatality_Cost")
  PDO_Cost = a_rst("PDO_Cost")

  'Look up accident rates from lookup table
  Fat_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_auto)
  If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
  End If
  Inj_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Injury, veht_auto)
  If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
  End If
  Prop_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Property, veht_auto)
  If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
  End If

'Accident reduction due to shifted vmt
  auto_Fatality_i = auto_vmt_shift_i * Fat_Rate_Auto / 1000000
  auto_Fatality_f = auto_vmt_shift_f * Fat_Rate_Auto / 1000000
  auto_Injury_i = auto_vmt_shift_i * Inj_Rate_Auto / 1000000
  auto_Injury_f = auto_vmt_shift_f * Inj_Rate_Auto / 1000000
  auto_Property_i = auto_vmt_shift_i * Prop_Rate_Auto / 1000000
  auto_Property_f = auto_vmt_shift_f * Prop_Rate_Auto / 1000000
  m_Fatality = (auto_Fatality_i + auto_Fatality_f) * (Forecast_Period) / 2
  m_Injury = (auto_Injury_i + auto_Injury_f) * (Forecast_Period) / 2
  m_Property = (auto_Property_i + auto_Property_f) * (Forecast_Period) / 2

  'Calculate safety benefits
  Safety_Ben_i = (-1) * ((auto_Fatality_i * Fatality_Cost) + _
    (auto_Injury_i * Evident_Cost) + (auto_Property_i * PDO_Cost))
  Safety_Ben_f = (-1) * ((auto_Fatality_f * Fatality_Cost) + _
    (auto_Injury_f * Evident_Cost) + (auto_Property_f * PDO_Cost))
  G_safety = (Safety_Ben_f - Safety_Ben_i) / (Forecast_Period - 1)
  m_Safety_Ben = (Safety_Ben_i * PofA) + (G_safety * PofG)

'Set module level values (to be displayed)
  m_Values.Add m_Fatality, "FATALITY"
  m_Values.Add m_Injury, "INJURY"
  m_Values.Add m_Property, "PROPERTY"
  m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
'Capital Cost
For i = 1 To 5
  m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
  m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _

```



```

    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
    m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

    Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
    m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations

```

```

    Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
    If (m_Total_Cost - m_Terminal_Cost) = 0 Then
        m_BCR = 0
    Else
        m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
    End If
    If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
        m_WSDOT_BCR = 0
    Else
        m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
    End If
    'Set module level values (to be displayed)
    m_Values.Add m_Total_Cost, "TOTAL_COST"
    m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
    m_Values.Add m_Federal_Cost, "FEDERAL_COST"
    m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
    m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
    m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
    m_Values.Add m_Other1_Cost, "OTHER1_COST"
    m_Values.Add m_Other2_Cost, "OTHER2_COST"
    m_Values.Add m_Other3_Cost, "OTHER3_COST"
    m_Values.Add m_Cap_Cost, "CAP_COST"
    m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
    m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
    m_Values.Add m_Other1_Cap, "OTHER1_CAP"
    m_Values.Add m_Other2_Cap, "OTHER2_CAP"
    m_Values.Add m_Other3_Cap, "OTHER3_CAP"
    m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
    m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
    m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
    m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
    m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
    m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
    m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
    m_Values.Add m_WSDOT_OM, "WSDOT_OM"
    m_Values.Add m_Federal_OM, "FEDERAL_OM"
    m_Values.Add m_Other1_OM, "OTHER1_OM"
    m_Values.Add m_Other2_OM, "OTHER2_OM"
    m_Values.Add m_Other3_OM, "OTHER3_OM"
    m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
    m_Values.Add m_BCR, "BCR"
    m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

'Calculation for the System Operation and Maintenance

m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)

m_Values.Add m_Sys_OM, "SYS_OM"

'Calculation for the System Preservation

m_Sys_Pres = 100 * m_rst("Q2A")

m_Values.Add m_Sys_Pres, "SYS_PRES"

'Calculation for the Special Needs Transportation

```

    m_Sp_Needs = 100 * m_rst("Q3A")
    m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
  Else
    m_Cong_Rel = (m_rst("Q4") * 50)
  End If
  m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
  Else
    m_Safety = m_rst("Q7B") * 50
  End If
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  'The -1 Multiplication Corrects the negative sign introduced for true
  m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
  Else
    m_Collab = (m_rst("Q10A") * 50) + 50
  End If
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
  If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + m_rst("Q14A") * 50
  Else
    m_Air_Qual = 0
  End If
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _

```

```
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub
```

Track Improvement

Option Compare Database

'Variables for the class level database objects

```
Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection
```

'Variables used in intermediate calculations (variants)

```
Dim Forecast_Period
Dim m_Pers_TT_Sav
Dim m_Pers_TT_Ben
Dim m_Freight_TT_Sav
Dim m_Freight_TT_Ben
Dim m_TT_Sav
Dim m_OpCost
Dim m_Fare_Cost
Dim m_VMT_Shift
Dim m_Approach
Dim m_WSDOT_Cap
Dim m_Federal_Cap
Dim m_Other1_Cap
Dim m_Other2_Cap
Dim m_Other3_Cap
Dim m_WSDOT_OM
Dim m_Federal_OM
Dim m_Other1_OM
Dim m_Other2_OM
Dim m_Other3_OM
Dim m_FishBarrier_Ben
Dim m_StormWater_Ben
Dim m_NoiseBarrier_Ben
Dim m_FishBarrier_Cap
Dim m_StormWater_Cap
Dim m_NoiseBarrier_Cap
```

'Variables to hold the calculated values (variants)

```
Dim m_Total_Benefit
Dim m_TT_Min
Dim m_TT_Ben
Dim m_User_Ben
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
```

```

Dim m_EnvRetrofit_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
Dim m_Other2_Cost
Dim m_Other3_Cost
Dim m_Cap_Cost
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR

```

'Variables to hold the outcome objective scores (variants)

```

Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource

```

```

'Speed Assumed for lookup table
Private Const AUTOSPEED = 50
Private Const TRUCKSPEED = 50

```

```

Private Sub Class_Initialize()
    m_qryName = "rail_calc_Track Impr"
    m_calcsComplete = False
End Sub

```

Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean

```

    Dim atype As String
    Dim qryDef As DAO.QueryDef

```

```

    Set m_dbs = CurrentDb

```

```

    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid

```

```

    ' Set recordset to new values.

```

```

    Set m_rst = qryDef.OpenRecordset

```

```

    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then

```

```

'Initialize new values collection to store calculated values
Set m_Values = New ValueCollection

'If you are running MICA use the scenario assumptions
If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
    atype = "prj_Project_Assumptions"
Else
    atype = "prj_Global_Assumptions"
    pid = asmptnID
End If

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
    Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
    Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
    Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")

'Calculate economic analysis factors
    Discount_Rate = a_rst("Discount_Rate")
    PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
    PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _

```

$$\text{PofF} = (1 + \text{Discount_Rate})^{(-\text{Forecast_Period})} - \frac{\text{Forecast_Period}}{\text{Discount_Rate} \cdot ((1 + \text{Discount_Rate})^{\text{Forecast_Period}} - 1)}$$

'Travel Time Savings Calculations

'Passenger Travel Time Savings

```

ann_pass_BCI = m_rst("ann_pass_BCI")
ann_pass_BCF = m_rst("ann_pass_BCF")
ann_pass_PCI = m_rst("ann_pass_PCI")
ann_pass_PCF = m_rst("ann_pass_PCF")
Time_Value_Pers = a_rst("Time_Value_Pers")

If m_rst("avg_speed_pass_BC") <> 0 Then
  tt_trip_bc = m_rst("pass_trip_length_BC") / m_rst("avg_speed_pass_BC")
End If
If m_rst("avg_speed_pass_PC") <> 0 Then
  tt_trip_pc = m_rst("pass_trip_length_PC") / m_rst("avg_speed_pass_PC")
End If

tt_pers_sav_i = (tt_trip_bc - tt_trip_pc) * ann_pass_BCI
tt_pers_sav_f = (tt_trip_bc - tt_trip_pc) * ann_pass_BCF
m_Pers_TT_Sav = (tt_pers_sav_i + tt_pers_sav_f) * Forecast_Period / 2

tt_ben_i = tt_pers_sav_i * Time_Value_Pers
tt_ben_f = tt_pers_sav_f * Time_Value_Pers

G_tt_ben = (tt_ben_f - tt_ben_i) / (Forecast_Period - 1)
m_Pers_TT_Ben = (tt_ben_i * PofA) + (G_tt_ben * PofG)

```

'Freight Travel Time Savings

```

Time_Value_Freight_Rail = a_rst("Time_Value_Freight_Rail")

freight_train_wk_BCI = m_rst("freight_train_wk_BCI")
freight_train_wk_BCF = m_rst("freight_train_wk_BCF")
freight_train_wk_PCI = m_rst("freight_train_wk_PCI")
freight_train_wk_PCF = m_rst("freight_train_wk_PCF")

tt_freight_bc = m_rst("freight_trip_length_BC") / m_rst("avg_speed_freight_BC")
tt_freight_pc = m_rst("freight_trip_length_PC") / m_rst("avg_speed_freight_PC")

tt_freight_sav_i = (tt_freight_bc - tt_freight_pc) * freight_train_wk_BCI * 52
tt_freight_sav_f = (tt_freight_bc - tt_freight_pc) * freight_train_wk_BCF * 52
m_Freight_TT_Sav = (tt_freight_sav_i + tt_freight_sav_f) * Forecast_Period / 2

tt_frt_ben_i = tt_freight_sav_i * Time_Value_Freight_Rail
tt_frt_ben_f = tt_freight_sav_f * Time_Value_Freight_Rail

G_tt_frt_ben = (tt_frt_ben_f - tt_frt_ben_i) / (Forecast_Period - 1)
m_Freight_TT_Ben = (tt_frt_ben_i * PofA) + (G_tt_frt_ben * PofG)

```

'Total Travel Time Calculations

```

m_TT_Sav = m_Pers_TT_Sav + m_Freight_TT_Sav
m_TT_Min = m_TT_Sav * 60

```



```
m_TT_Ben = m_Pers_TT_Ben + m_Freight_TT_Ben
```

```
'Person shift
```

```
pers_shift_i = ann_pass_PCI - ann_pass_BCI
```

```
pers_shift_f = ann_pass_PCF - ann_pass_BCF
```

```
m_Pers_Shift = (pers_shift_i + pers_shift_f) * Forecast_Period / 2
```

```
'Auto VMT shift
```

```
avo = a_rst("avo")
```

```
auto_divert_rail = m_rst("auto_divert_rail")
```

```
auto_vmt_shift_i = pers_shift_i * auto_divert_rail / avo * (-1)
```

```
auto_vmt_shift_f = pers_shift_f * auto_divert_rail / avo * (-1)
```

```
auto_vmt_shift = (auto_vmt_shift_i + auto_vmt_shift_f) * Forecast_Period / 2
```

```
'Shift from freight trucks to rail
```

```
general_tonpertruck = a_rst("general_tonpertruck")
```

```
freight_trip_length_BC = m_rst("freight_trip_length_BC")
```

```
freight_trip_length_PC = m_rst("freight_trip_length_PC")
```

```
truck_rem_i = (m_rst("ton_freight_BCI") - m_rst("ton_freight_PCI")) / _  
general_tonpertruck
```

```
truck_rem_f = (m_rst("ton_freight_BCF") - m_rst("ton_freight_PCF")) / _  
general_tonpertruck
```

```
m_Truck_Rem = (truck_rem_i + truck_rem_f) * Forecast_Period / 2
```

```
'Truck VMT shift
```

```
truck_vmt_shift_i = truck_rem_i * freight_trip_length_BC
```

```
truck_vmt_shift_f = truck_rem_f * freight_trip_length_BC
```

```
truck_vmt_shift = (truck_vmt_shift_i + truck_vmt_shift_f) * Forecast_Period / 2
```

```
m_VMT_Shift = auto_vmt_shift + truck_vmt_shift
```

```
'Added Rail VMT
```

```
rail_vmt_add_i = ((freight_train_wk_PCI * freight_trip_length_PC) - _  
(freight_train_wk_BCI * freight_trip_length_BC)) * 52
```

```
rail_vmt_add_f = ((freight_train_wk_PCF * freight_trip_length_PC) - _  
(freight_train_wk_BCF * freight_trip_length_BC)) * 52
```

```
m_Rail_Vmt_Add = (rail_vmt_add_i + rail_vmt_add_f) * Forecast_Period / 2
```

```
'Set module level values (to be displayed)
```

```
m_Values.Add Forecast_Period, "FORECAST_PERIOD"
```

```
m_Values.Add m_Pers_TT_Sav, "PERS_TT_SAV"
```

```
m_Values.Add m_Pers_TT_Ben, "PERS_TT_BEN"
```

```
m_Values.Add m_Freight_TT_Sav, "FREIGHT_TT_SAV"
```

```
m_Values.Add m_Freight_TT_Ben, "FREIGHT_TT_BEN"
```

```
m_Values.Add m_TT_Sav, "TT_SAV"
```

```
m_Values.Add m_TT_Ben, "TT_BEN"
```

```
m_Values.Add m_Pers_Shift, "PERS_SHIFT"
```

```
m_Values.Add m_Truck_Rem, "TRUCK_REM"
```

```
m_Values.Add m_VMT_Shift, "VMT_SHIFT"
```

```
m_Values.Add m_Rail_Vmt_Add, "RAIL_VMT_ADD"
```

```
'Operating Cost Savings Calculations
```

```

Veh_OpCost_Full = a_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = a_rst("Veh_OpCost_Direct")
Truck_OpCost_Full = a_rst("Truck_OpCost_Full")
Truck_OpCost_Direct = a_rst("Truck_OpCost_Direct")
If a_rst("Full_Cost") Then
  m_Approach = "Full cost"
  OpCost_i = ((auto_vmt_shift_i * Veh_OpCost_Full) + _
    (truck_vmt_shift_i * Truck_OpCost_Full)) * (-1)
  OpCost_f = ((auto_vmt_shift_f * Veh_OpCost_Full) + _
    (truck_vmt_shift_f * Truck_OpCost_Full)) * (-1)
Else
  m_Approach = "Direct cost"
  OpCost_i = ((auto_vmt_shift_i * Veh_OpCost_Direct) + _
    (truck_vmt_shift_i * Truck_OpCost_Direct)) * (-1)
  OpCost_f = ((auto_vmt_shift_f * Veh_OpCost_Direct) + _
    (truck_vmt_shift_f * Truck_OpCost_Direct)) * (-1)
End If

G_OpCost = (OpCost_f - OpCost_i) / (Forecast_Period - 1)
m_OpCost = (OpCost_i * PofA) + (G_OpCost * PofG)

'Out-of-pocket costs to users due to fares
Fare = m_rst("fare")
fare_cost_i = pers_shift_i * Fare * (-1)
fare_cost_f = pers_shift_f * Fare * (-1)
G_fare_cost = (fare_cost_f - fare_cost_i) / (Forecast_Period - 1)
m_Fare_Cost = (fare_cost_i * PofA) + (G_fare_cost * PofG)

'Pavement Cost Savings
truck_pave_cost = a_rst("truck_pave_cost")
Pavement_Sav_i = (-1) * truck_vmt_shift_i * truck_pave_cost
Pavement_Sav_f = (-1) * truck_vmt_shift_f * truck_pave_cost

G_Pavement_Sav = (Pavement_Sav_f - Pavement_Sav_i) / (Forecast_Period - 1)
m_Pavement_Sav = (Pavement_Sav_i * PofA) + (G_Pavement_Sav * PofG)

'User Benefit NPV Calculation
m_User_Ben = m_OpCost + m_Fare_Cost + m_Pavement_Sav

'Set module level values (to be displayed)
m_Values.Add m_Approach, "APPROACH"
m_Values.Add m_OpCost, "OPCOST"
m_Values.Add m_Fare_Cost, "FARE_COST"
m_Values.Add m_Pavement_Sav, "PAVEMENT_SAV"
m_Values.Add m_User_Ben, "USER_BEN"

'Air Pollution Calculations - for diverted auto trips
COTon_Cost = a_rst("COTon_Cost")
VOCTon_Cost = a_rst("VOCTon_Cost")
NOXTon_Cost = a_rst("NOXTon_Cost")
PM10Ton_Cost = a_rst("PM10Ton_Cost")

CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)
If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If

```

```

VOC_Rate_Auto = a_rst("VOC_Rate_Auto")
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")

CO_Rate_Truck = LookupAirPollution(m_dbs, TRUCKSPEED, em_CO, veht_Truck)
If a_rst("CO_Rate_Truck") <> CO_Rate_Truck Or IsNull(a_rst("CO_Rate_Truck")) Then
    Call UpdateRstField(a_rst, "CO_Rate_Truck", CO_Rate_Truck)
End If
VOC_Rate_Truck = a_rst("VOC_Rate_Truck")
NOX_Rate_Truck = LookupAirPollution(m_dbs, TRUCKSPEED, em_NOX, veht_Truck)
If a_rst("NOX_Rate_Truck") <> NOX_Rate_Truck Or IsNull(a_rst("NOX_Rate_Truck")) Then
    Call UpdateRstField(a_rst, "NOX_Rate_Truck", NOX_Rate_Truck)
End If
PM10_Rate_Truck = a_rst("PM10_Rate_Truck")

CO_Rate_Rail = a_rst("CO_Rate_Rail")
VOC_Rate_Rail = a_rst("VOC_Rate_Rail")
NOX_Rate_Rail = a_rst("NOX_Rate_Rail")
PM10_Rate_Rail = a_rst("PM10_Rate_Rail")

CO_Tons_i = ((truck_vmt_shift_i * CO_Rate_Truck) + _
    (auto_vmt_shift_i * CO_Rate_Auto) - (rail_vmt_add_i * CO_Rate_Rail)) * _
    (1 / 1000) * (0.9842 / 1000)
CO_Tons_f = ((truck_vmt_shift_f * CO_Rate_Truck) + _
    (auto_vmt_shift_f * CO_Rate_Auto) - (rail_vmt_add_f * CO_Rate_Rail)) * _
    (1 / 1000) * (0.9842 / 1000)
VOC_Tons_i = ((truck_vmt_shift_i * VOC_Rate_Truck) + _
    (auto_vmt_shift_i * VOC_Rate_Auto) - (rail_vmt_add_i * VOC_Rate_Rail)) * _
    (1 / 1000) * (0.9842 / 1000)
VOC_Tons_f = ((truck_vmt_shift_f * VOC_Rate_Truck) + _
    (auto_vmt_shift_f * VOC_Rate_Auto) - (rail_vmt_add_f * VOC_Rate_Rail)) * _
    (1 / 1000) * (0.9842 / 1000)
NOX_Tons_i = ((truck_vmt_shift_i * NOX_Rate_Truck) + _
    (auto_vmt_shift_i * NOX_Rate_Auto) - (rail_vmt_add_i * NOX_Rate_Rail)) * _
    (1 / 1000) * (0.9842 / 1000)
NOX_Tons_f = ((truck_vmt_shift_f * NOX_Rate_Truck) + _
    (auto_vmt_shift_f * NOX_Rate_Auto) - (rail_vmt_add_f * NOX_Rate_Rail)) * _
    (1 / 1000) * (0.9842 / 1000)
PM10_Tons_i = ((truck_vmt_shift_i * PM10_Rate_Truck) + _
    (auto_vmt_shift_i * PM10_Rate_Auto) - (rail_vmt_add_i * PM10_Rate_Rail)) * _
    (1 / 1000) * (0.9842 / 1000)
PM10_Tons_f = ((truck_vmt_shift_f * PM10_Rate_Truck) + _
    (auto_vmt_shift_f * PM10_Rate_Auto) - (rail_vmt_add_f * PM10_Rate_Rail)) * _
    (1 / 1000) * (0.9842 / 1000)

'Emission Sums
m_CO_Tons = (CO_Tons_f + CO_Tons_i) * Forecast_Period / 2
m_VOC_Tons = (VOC_Tons_f + VOC_Tons_i) * Forecast_Period / 2
m_NOX_Tons = (NOX_Tons_f + NOX_Tons_i) * Forecast_Period / 2
m_PM10_Tons = (PM10_Tons_f + PM10_Tons_i) * Forecast_Period / 2

'Emission Benefit Calculations
EnvBen_i = (CO_Tons_i * COTon_Cost + VOC_Tons_i * VOCTon_Cost _

```

```

+ NOX_Tons_i * NOXTon_Cost + PM10_Tons_i * PM10Ton_Cost) * (-1)
EnvBen_f = (CO_Tons_f * COTon_Cost + VOC_Tons_f * VOCTon_Cost _
+ NOX_Tons_f * NOXTon_Cost + PM10_Tons_f * PM10Ton_Cost) * (-1)
G_EnvBen = (EnvBen_f - EnvBen_i) / (Forecast_Period - 1)

```

'Emission NPV Calculations

```
m_Env_Ben = (EnvBen_i * PofA) + (G_EnvBen * PofG)
```

'Set module level values (to be displayed)

```

m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"
m_Values.Add m_PM10_Tons, "PM10_TONS"
m_Values.Add m_Env_Ben, "ENV_BEN"

```

'Safety Calculations

```
Evident_Cost = a_rst("Evident_Cost")
```

```
Fatality_Cost = a_rst("Fatality_Cost")
```

```
PDO_Cost = a_rst("PDO_Cost")
```

```
Fat_Rate_Rail = a_rst("Fat_Rate_Rail")
```

```
Inj_Rate_Rail = a_rst("Inj_Rate_Rail")
```

```
Prop_Rate_Rail = a_rst("Prop_Rate_Rail")
```

'Look up auto accident rates from lookup table

```
Fat_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_auto)
```

```
If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
```

```
End If
```

```
Inj_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Injury, veht_auto)
```

```
If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
```

```
End If
```

```
Prop_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Property, veht_auto)
```

```
If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
```

```
End If
```

'Look up truck accident rates from lookup table

```
Fat_Rate_Truck = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_Truck)
```

```
If a_rst("Fat_Rate_Truck") <> Fat_Rate_Truck Or IsNull(a_rst("Fat_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Fat_Rate_Truck", Fat_Rate_Truck)
```

```
End If
```

```
Inj_Rate_Truck = LookupFatalityRate(m_dbs, 2, s_Injury, veht_Truck)
```

```
If a_rst("Inj_Rate_Truck") <> Inj_Rate_Truck Or IsNull(a_rst("Inj_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Inj_Rate_Truck", Inj_Rate_Truck)
```

```
End If
```

```
Prop_Rate_Truck = LookupFatalityRate(m_dbs, 2, s_Property, veht_Truck)
```

```
If a_rst("Prop_Rate_Truck") <> Prop_Rate_Truck Or IsNull(a_rst("Prop_Rate_Truck")) Then
  Call UpdateRstField(a_rst, "Prop_Rate_Truck", Prop_Rate_Truck)
```

```
End If
```

'Accident reduction due to shifted vmt

```
auto_Fatality_i = auto_vmt_shift_i * Fat_Rate_Auto / 1000000
```

```
auto_Fatality_f = auto_vmt_shift_f * Fat_Rate_Auto / 1000000
```

```

auto_Injury_i = auto_vmt_shift_i * Inj_Rate_Auto / 1000000
auto_Injury_f = auto_vmt_shift_f * Inj_Rate_Auto / 1000000
auto_Property_i = auto_vmt_shift_i * Prop_Rate_Auto / 1000000
auto_Property_f = auto_vmt_shift_f * Prop_Rate_Auto / 1000000

```

```

truck_Fatality_i = truck_vmt_shift_i * Fat_Rate_Truck / 1000000
truck_Fatality_f = truck_vmt_shift_f * Fat_Rate_Truck / 1000000
truck_Injury_i = truck_vmt_shift_i * Inj_Rate_Truck / 1000000
truck_Injury_f = truck_vmt_shift_f * Inj_Rate_Truck / 1000000
truck_Property_i = truck_vmt_shift_i * Prop_Rate_Truck / 1000000
truck_Property_f = truck_vmt_shift_f * Prop_Rate_Truck / 1000000

```

```

rail_Fatality_i = rail_vmt_add_i * Fat_Rate_Rail / 1000000
rail_Fatality_f = rail_vmt_add_f * Fat_Rate_Rail / 1000000
rail_Injury_i = rail_vmt_add_i * Inj_Rate_Rail / 1000000
rail_Injury_f = rail_vmt_add_f * Inj_Rate_Rail / 1000000
rail_Property_i = rail_vmt_add_i * Prop_Rate_Rail / 1000000
rail_Property_f = rail_vmt_add_f * Prop_Rate_Rail / 1000000

```

'Accident reduction due to shifted vmt

```

m_Fatality = ((auto_Fatality_i + truck_Fatality_i - rail_Fatality_i) + _
(auto_Fatality_f + truck_Fatality_f - rail_Fatality_f)) * _
(Forecast_Period) / 2
m_Injury = ((auto_Injury_i + truck_Injury_i - rail_Injury_i) + _
(auto_Injury_f + truck_Injury_f - rail_Injury_f)) * _
(Forecast_Period) / 2
m_Property = ((auto_Property_i + truck_Property_i - rail_Property_i) + _
(auto_Property_f + truck_Property_f - rail_Property_f)) * _
(Forecast_Period) / 2

```

'Calculate safety benefits

```

Safety_Ben_i = (((auto_Fatality_i + truck_Fatality_i - rail_Fatality_i) * Fatality_Cost) + _
((auto_Injury_i + truck_Injury_i - rail_Injury_i) * Evident_Cost) + _
((auto_Property_i + truck_Property_i - rail_Property_i) * PDO_Cost)) * (-1)
Safety_Ben_f = (((auto_Fatality_f + truck_Fatality_f - rail_Fatality_f) * Fatality_Cost) + _
((auto_Injury_f + truck_Injury_f - rail_Injury_f) * Evident_Cost) + _
((auto_Property_f + truck_Property_f - rail_Property_f) * PDO_Cost)) * (-1)
G_safety = (Safety_Ben_f - Safety_Ben_i) / (Forecast_Period - 1)
m_Safety_Ben = (Safety_Ben_i * PofA) + (G_safety * PofG)

```

'Set module level values (to be displayed)

```

m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

```

'Cost Calculations

'Capital Cost

For i = 1 To 5

```

m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _

```

```

    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
    m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

    Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

'Environmental Retrofit Costs

For i = 1 To 5
    m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

'Environmental Retrofit Benefits
m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

'Benefit-Cost Calculations

```

```

    Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
    If (m_Total_Cost - m_Terminal_Cost) = 0 Then
        m_BCR = 0
    Else
        m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
    End If
    If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
        m_WSDOT_BCR = 0
    Else
        m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
    End If
    'Set module level values (to be displayed)
    m_Values.Add m_Total_Cost, "TOTAL_COST"
    m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
    m_Values.Add m_Federal_Cost, "FEDERAL_COST"
    m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
    m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
    m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
    m_Values.Add m_Other1_Cost, "OTHER1_COST"
    m_Values.Add m_Other2_Cost, "OTHER2_COST"
    m_Values.Add m_Other3_Cost, "OTHER3_COST"
    m_Values.Add m_Cap_Cost, "CAP_COST"
    m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
    m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
    m_Values.Add m_Other1_Cap, "OTHER1_CAP"
    m_Values.Add m_Other2_Cap, "OTHER2_CAP"
    m_Values.Add m_Other3_Cap, "OTHER3_CAP"
    m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
    m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
    m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
    m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
    m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
    m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
    m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
    m_Values.Add m_WSDOT_OM, "WSDOT_OM"
    m_Values.Add m_Federal_OM, "FEDERAL_OM"
    m_Values.Add m_Other1_OM, "OTHER1_OM"
    m_Values.Add m_Other2_OM, "OTHER2_OM"
    m_Values.Add m_Other3_OM, "OTHER3_OM"
    m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
    m_Values.Add m_BCR, "BCR"
    m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next
    'Calculation for the System Operation and Maintenance
    m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
    m_Values.Add m_Sys_OM, "SYS_OM"
    'Calculation for the System Preservation
    m_Sys_Pres = 100 * m_rst("Q2A")
    m_Values.Add m_Sys_Pres, "SYS_PRES"
    'Calculation for the Special Needs Transportation

```

```

    m_Sp_Needs = 100 * m_rst("Q3A")
    m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
  Else
    m_Cong_Rel = (m_rst("Q4") * 50)
  End If
  m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
  Else
    m_Safety = m_rst("Q7B") * 50
  End If
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  'The -1 Multiplication Corrects the negative sign introduced for true
  m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
  Else
    m_Collab = (m_rst("Q10A") * 50) + 50
  End If
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
  If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + m_rst("Q14A") * 50
  Else
    m_Air_Qual = 0
  End If
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _

```



```
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat`
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

End Sub

Public Function GetItemValue(itemname As String) As Variant
    Dim rtnval

    rtnval = m_Values.Retrieve(itemname)

    GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub
```

Program Code for Transit Projects

STEAM Calculations

Option Compare Database

'Variables for the class level database objects

Dim m_dbs As DAO.Database

Dim m_rst As DAO.Recordset

Dim m_Values As Collection

'Variables to hold the calculated values (variants)

Dim m_TT_Min

Dim m_TT_Ben

Dim m_User_Ben

Dim m_User_Transfer

Dim m_CO_Tons

Dim m_VOC_Tons

Dim m_NOX_Tons

Dim m_PM10_Tons

Dim m_Env_Ben

Dim m_Fatality

Dim m_Injury

Dim m_Property

Dim m_Safety_Ben

Dim m_Total_Cost

Dim m_WSDOT_Cost

Dim m_Federal_Cost

Dim m_Terminal_Cost

Dim m_BCR

Dim m_WSDOT_BCR

'Variables to hold the outcome objective scores (variants)

Dim m_Sys_OM

Dim m_Sys_Pres

Dim m_Sp_Needs

Dim m_Cong_Rel

Dim m_Trav_Opt

Dim m_Seamless

Dim m_Safety

Dim m_Security

Dim m_Commnty

Dim m_Collab

Dim m_Freight

Dim m_Econ_Proc

Dim m_Tourism

Dim m_Air_Qual

Dim m_Wtr_Qual

Dim m_Habitat

Dim m_Resource

'Speed Assumed for lookup table

Private Const AUTOSPEED = 35

Public Function SetProjectNumber(pid As Long) As Boolean

```

Dim rtnval As Boolean
Dim qryDef As DAO.QueryDef

'Initialize return value to false - set true if one record is returned
rtnval = False

Set m_dbs = CurrentDb

' Open QueryDef object with one parameters.
Set qryDef = m_dbs.QueryDefs("tran_calc_STEAM")
qryDef.Parameters!projID = pid

' Set recordset to new values.
Set m_rst = qryDef.OpenRecordset
' If it is a unique data entry then calculate all scores
If Not m_rst.EOF Then
    Set m_Values = New Collection
    CalculateBenefits
    CalculateOutObjScores
    rtnval = True
End If
SetProjectNumber = rtnval
End Function

```

```
Private Sub CalculateBenefits()
```

```

'In Vehicle Travel Time Calculations:
Ann_Daily_Benefit = m_rst("Ann_Daily_Benefit")
InVehAuto_PCI = m_rst("InVehAuto_PCI")
InVehAuto_BCI = m_rst("InVehAuto_BCI")
InVehAuto_PCF = m_rst("InVehAuto_PCF")
InVehAuto_BCF = m_rst("InVehAuto_BCF")
InVehFr_PCI = m_rst("InVehFr_PCI")
InVehFr_BCI = m_rst("InVehFr_BCI")
InVehFr_PCF = m_rst("InVehFr_PCF")
InVehFr_BCF = m_rst("InVehFr_BCF")
InVehBus_PCI = m_rst("InVehBus_PCI")
InVehBus_BCI = m_rst("InVehBus_BCI")
InVehBus_PCF = m_rst("InVehBus_PCF")
InVehBus_BCF = m_rst("InVehBus_BCF")
InVehRail_PCI = m_rst("InVehRail_PCI")
InVehRail_BCI = m_rst("InVehRail_BCI")
InVehRail_PCF = m_rst("InVehRail_PCF")
InVehRail_BCF = m_rst("InVehRail_BCF")
OutVehAuto_PCI = m_rst("OutVehAuto_PCI")
OutVehAuto_BCI = m_rst("OutVehAuto_BCI")
OutVehAuto_PCF = m_rst("OutVehAuto_PCF")
OutVehAuto_BCF = m_rst("OutVehAuto_BCF")
OutVehFr_PCI = m_rst("OutVehFr_PCI")
OutVehFr_BCI = m_rst("OutVehFr_BCI")
OutVehFr_PCF = m_rst("OutVehFr_PCF")
OutVehFr_BCF = m_rst("OutVehFr_BCF")
OutVehBus_PCI = m_rst("OutVehBus_PCI")
OutVehBus_BCI = m_rst("OutVehBus_BCI")

```

```

OutVehBus_PCF = m_rst("OutVehBus_PCF")
OutVehBus_BCF = m_rst("OutVehBus_BCF")
OutVehRail_PCI = m_rst("OutVehRail_PCI")
OutVehRail_BCI = m_rst("OutVehRail_BCI")
OutVehRail_PCF = m_rst("OutVehRail_PCF")
OutVehRail_BCF = m_rst("OutVehRail_BCF")

'In Vehicle Travel Time Calculations:
'Yearly Travel Time Benefits in Minutes
  InVehAuto_I = (InVehAuto_PCI - InVehAuto_BCI) * 60 * 1000000
  InVehFr_I = (InVehFr_PCI - InVehFr_BCI) * 60 * 1000000
  InVehBus_I = (InVehBus_PCI - InVehBus_BCI) * 60 * 1000000
  InVehRail_I = (InVehRail_PCI - InVehRail_BCI) * 60 * 1000000
  InVehAuto_F = (InVehAuto_PCF - InVehAuto_BCF) * 60 * 1000000
  InVehFr_F = (InVehFr_PCF - InVehFr_BCF) * 60 * 1000000
  InVehBus_F = (InVehBus_PCF - InVehBus_BCF) * 60 * 1000000
  InVehRail_F = (InVehRail_PCF - InVehRail_BCF) * 60 * 1000000
'Time value the same for auto, bus, and rail travel
  InVeh_I = InVehAuto_I + InVehBus_I + InVehRail_I
  InVeh_F = InVehAuto_F + InVehBus_F + InVehRail_F
'Out of Vehicle Travel Time Calculations:
'Yearly Travel Time Benefits in Minutes
  OutVehAuto_I = (OutVehAuto_PCI - OutVehAuto_BCI)
  OutVehFr_I = (OutVehFr_PCI - OutVehFr_BCI)
  OutVehBus_I = (OutVehBus_PCI - OutVehBus_BCI)
  OutVehRail_I = (OutVehRail_PCI - OutVehRail_BCI)
  OutVehAuto_F = (OutVehAuto_PCF - OutVehAuto_BCF)
  OutVehFr_F = (OutVehFr_PCF - OutVehFr_BCF)
  OutVehBus_F = (OutVehBus_PCF - OutVehBus_BCF)
  OutVehRail_F = (OutVehRail_PCF - OutVehRail_BCF)
'Time value the same for auto, bus, and rail travel
  OutVeh_I = OutVehAuto_I + OutVehBus_I + OutVehRail_I
  OutVeh_F = OutVehAuto_F + OutVehBus_F + OutVehRail_F

  TT_Min_I = (InVeh_I + InVehFr_I + OutVeh_I + OutVehFr_I)
  TT_Min_F = (InVeh_F + InVehFr_F + OutVeh_F + OutVehFr_F)
  FrTT_Min_I = (InVehFr_I + OutVehFr_I)
  FrTT_Min_F = (InVehFr_F + OutVehFr_F)

  N = m_rst("Fore_Year") - m_rst("Init_Year") + 1
  If IsNumeric(TT_Min_F) And IsNumeric(TT_Min_I) And IsNumeric(N) And N > 0 Then
    LogTT_Min = Log((TT_Min_F / TT_Min_I))
    NPVF_Min = (Exp(((LogTT_Min / N)) * N) - 1) / _
              ((LogTT_Min / N))
  Else
    NPVF_Min = Null
  End If
  If IsNumeric(FrTT_Min_F) And IsNumeric(FrTT_Min_I) And IsNumeric(N) And N > 0 Then
    LogTT_FrMin = Log((FrTT_Min_F / FrTT_Min_I))
    NPVF_FrMin = (Exp(((LogTT_FrMin / N)) * N) - 1) / _
              ((LogTT_FrMin / N))
  Else
    NPVF_FrMin = Null
  End If
  m_TT_Min = TT_Min_I * NPVF_Min
  FrTT_Min = FrTT_Min_I * NPVF_FrMin

```

```

'Set module level values (to be displayed)
  m_Values.Add m_TT_Min, "TT_MIN"
  m_Values.Add FrTT_Min, "FrTT_MIN"
'Number of Trips
TripsAuto_I = (TripsAuto_PCI - TripsAuto_BCI) * 1000000
TripsFr_I = (TripsFr_PCI - TripsFr_BCI) * 1000000
TripsBus_I = (TripsBus_PCI - TripsBus_BCI) * 1000000
TripsRail_I = (TripsRail_PCI - TripsRail_BCI) * 1000000
TripsAuto_F = (TripsAuto_PCF - TripsAuto_BCF) * 1000000
TripsFr_F = (TripsFr_PCF - TripsFr_BCF) * 1000000
TripsBus_F = (TripsBus_PCF - TripsBus_BCF) * 1000000
TripsRail_F = (TripsRail_PCF - TripsRail_BCF) * 1000000
'Travel Time Benefits in Dollars
  TTAuto_Ben_I = ((InVeh_I) / 60 * (Percent_TV_InVeh * Time_Value_Veh)) + (OutVeh_I) / 60
* (Percent_TV_OutVeh * Time_Value_Veh)
  TTAuto_Ben_F = ((InVeh_F) / 60 * (Percent_TV_InVeh * Time_Value_Veh)) + (OutVeh_F) /
60 * (Percent_TV_OutVeh * Time_Value_Veh)
  FrTT_Ben_I = ((InVehFr_I) / 60 * (Percent_TV_InVeh * Time_Value_Freight)) +
(OutVehFr_I) / 60 * (Percent_TV_OutVeh * Time_Value_Freight)
  FrTT_Ben_F = ((InVehFr_F) / 60 * (Percent_TV_InVeh * Time_Value_Freight)) +
(OutVehFr_F) / 60 * (Percent_TV_OutVeh * Time_Value_Freight)
  TT_Ben_I = TTAuto_Ben_I + FrTT_Ben_I
  TT_Ben_F = TTAuto_Ben_F + FrTT_Ben_F
If IsNumeric(TT_Ben_F) And IsNumeric(TT_Ben_I) And IsNumeric(N) And N > 0 Then
  LogTT_Ben = Log((TT_Ben_F / TT_Ben_I))
  NPVF_TTBen = (Exp(((LogTT_Ben / N) - Discount_Rate) * N) - 1) / _
((LogTT_Ben / N) - Discount_Rate)
Else
  NPVF_TTBen = Null
End If
If IsNumeric(FrTT_Ben_F) And IsNumeric(FrTT_Ben_I) And IsNumeric(N) And N > 0 Then
  LogFrTT_Ben = Log((FrTT_Ben_F / FrTT_Ben_I))
  NPVF_FrTTBen = (Exp(((LogFrTT_Ben / N) - Discount_Rate) * N) - 1) / _
((LogFrTT_Ben / N) - Discount_Rate)
Else
  NPVF_TTBen = Null
End If
m_TT_Ben = TT_Ben_I * NPVF_TTBen
FrTT_Ben = FrTT_Ben_I * NPVF_FrTTBen
'Set module level values (to be displayed)
  m_Values.Add m_TT_Ben, "TT_BEN"
  m_Values.Add FrTT_Ben, "FrTT_BEN"

'Operating Cost Calculations
'Annual Change in VMT
VMTAuto_PCI = m_rst("VMTAuto_PCI")
VMTAuto_BCI = m_rst("VMTAuto_BCI")
VMTAuto_PCF = m_rst("VMTAuto_PCF")
VMTAuto_BCF = m_rst("VMTAuto_BCF")
VMTFr_PCI = m_rst("VMTFr_PCI")
VMTFr_BCI = m_rst("VMTFr_BCI")
VMTFr_PCF = m_rst("VMTFr_PCF")
VMTFr_BCF = m_rst("VMTFr_BCF")
VMTBus_PCI = m_rst("VMTBus_PCI")

```

```

VMTBus_BCI = m_rst("VMTBus_BCI")
VMTBus_PCF = m_rst("VMTBus_PCF")
VMTBus_BCF = m_rst("VMTBus_BCF")
VMTRail_PCI = m_rst("VMTRail_PCI")
VMTRail_BCI = m_rst("VMTRail_BCI")
VMTRail_PCF = m_rst("VMTRail_PCF")
VMTRail_BCF = m_rst("VMTRail_BCF")
VMTAuto_I = (VMTAuto_PCI - VMTAuto_BCI) * 1000000
FrVMT_I = (VMTFr_PCI - VMTFr_BCI) * 1000000
BusVMT_I = (VMTBus_PCI - VMTBus_BCI) * 1000000
RailVMT_I = (VMTRail_PCI - VMTRail_BCI) * 1000000
VMTAuto_F = (VMTAuto_PCF - VMTAuto_BCF) * 1000000
FrVMT_F = (VMTFr_PCF - VMTFr_BCF) * 1000000
BusVMT_F = (VMTBus_PCF - VMTBus_BCF) * 1000000
RailVMT_F = (VMTRail_PCF - VMTRail_BCF) * 1000000

If IsNumeric(VMTAuto_I) And IsNumeric(VMTAuto_F) And IsNumeric(N) And N > 0 Then
  LogVMTAuto = Log((VMTAuto_F / VMTAuto_I))
  NPVF_VMTAuto = ((Exp(LogVMTAuto) - 1) / (LogVMTAuto)) / N
Else
  NPVF_VMTAuto = Null
End If
VMTAuto_Tot = VMTAuto_I * NPVF_VMTAuto
If IsNumeric(FrVMT_F) And IsNumeric(FrVMT_I) And IsNumeric(N) And N > 0 Then
  LogFrVMT = Log((FrVMT_F / FrVMT_I))
  NPVF_FrVMT = ((Exp(LogFrVMT) - 1) / (LogFrVMT)) / N
Else
  NPVF_FrVMT = Null
End If
FrVMT_Tot = FrVMT_I * NPVF_FrVMT
If IsNumeric(BusVMT_F) And IsNumeric(BusVMT_I) And IsNumeric(N) And N > 0 Then
  LogBusVMT = Log((BusVMT_F / BusVMT_I))
  NPVF_BusVMT = ((Exp(LogBusVMT) - 1) / (LogBusVMT)) / N
Else
  NPVF_BusVMT = Null
End If
BusVMT_Tot = BusVMT_I * NPVF_BusVMT
If IsNumeric(RailVMT_F) And IsNumeric(RailVMT_I) And IsNumeric(N) And N > 0 Then
  LogRailVMT = Log((RailVMT_F / RailVMT_I))
  NPVF_RailVMT = ((Exp(LogRailVMT) - 1) / (LogRailVMT)) / N
Else
  NPVF_RailVMT = Null
End If

'User benefit cost calculations
Veh_OpCost_Full = m_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = m_rst("Veh_OpCost_Direct")
Freight_OpCost = m_rst("Freight_OpCost")
If m_rst("Full_Cost") Then
  UserBen_I = (VMTAuto_I * Veh_OpCost_Full) + (FrVMT_I * Freight_OpCost)
  UserBen_F = (VMTAuto_F * Veh_OpCost_Full) + (FrVMT_F * Freight_OpCost)
Else
  UserBen_I = (VMTAuto_I * Veh_OpCost_Direct) + (FrVMT_I * Freight_OpCost)
  UserBen_F = (VMTAuto_F * Veh_OpCost_Direct) + (FrVMT_F * Freight_OpCost)
End If
If IsNumeric(UserBen_F) And IsNumeric(UserBen_I) And IsNumeric(N) And N > 0 Then

```

```

    LogUserBen = Log((UserBen_F / UserBen_I))
    NPVF_UCBen = (Exp(((LogUserBen / N) - Discount_Rate) * N) - 1) / ((LogUserBen / N) -
Discount_Rate)
    Else
        NPVF_UCBen = Null
    End If
    m_User_Ben = UserBen_I * NPVF_UCBen
    'Set module level values (to be displayed)
    m_Values.Add m_User_Ben, "USER_BEN"

'Air Pollution
'Emissions Calculations
Per_Cold_Auto = m_rst("Per_Cold_Auto")
Per_Cold_Truck = m_rst("Per_Cold_Truck")
CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)
If m_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(m_rst("CO_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If
CO_Rate_Truck = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_truck)
If m_rst("CO_Rate_Truck") <> NOX_Rate_Truck Or IsNull(m_rst("CO_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "CO_Rate_Truck", CO_Rate_Truck)
End If
COTon_Cost = m_rst("COTon_Cost")
VOC_Rate_Auto = m_rst("VOC_Rate_Auto")
VOC_Rate_Truck = m_rst("VOC_Rate_Truck")
VOCTon_Cost = m_rst("VOCTon_Cost")
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
If m_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(m_rst("NOX_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
NOX_Rate_Truck = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_truck)
If m_rst("NOX_Rate_Truck") <> NOX_Rate_Truck Or IsNull(m_rst("NOX_Rate_Truck"))
Then
    Call UpdateRstField(m_rst, "NOX_Rate_Truck", NOX_Rate_Truck)
End If
NOXTon_Cost = m_rst("NOXTon_Cost")
PM10_Rate_Auto = m_rst("PM10_Rate_Auto")
PM10_Rate_Truck = m_rst("PM10_Rate_Truck")
PM10Ton_Cost = m_rst("PM10Ton_Cost")
CO_Rate_Rail = m_rst("CO_Rate_Rail")
VOC_Rate_Rail = m_rst("VOC_Rate_Rail")
NOX_Rate_Rail = m_rst("NOX_Rate_Rail")
PM10_Rate_Rail = m_rst("PM10_Rate_Rail")
CO_Rate_Bus = m_rst("CO_Rate_Bus")
VOC_Rate_Bus = m_rst("VOC_Rate_Bus")
NOX_Rate_Bus = m_rst("NOX_Rate_Bus")
PM10_Rate_Bus = m_rst("PM10_Rate_Bus")

CO_AutoTons = (VMTAuto_I * NPVF_VMTAuto * CO_Rate_Auto * (1 / 1000) * (0.9842 /
1000)) + (AutoTrips_I * NPVF_VMTAuto * Per_Cold_Auto * CO_Cold_Auto * (1 / 1000) * (0.9842
/ 1000))
VOC_AutoTons = (VMTAuto_I * NPVF_VMTAuto * VOC_Rate_Auto * (1 / 1000) * (0.9842 /
1000))
NOX_AutoTons = (VMTAuto_I * NPVF_VMTAuto * NOX_Rate_Auto * (1 / 1000) * (0.9842 /
1000)) + (AutoTrips_I * NPVF_VMTAuto * Per_Cold_Auto * VOC_Cold_Auto * (1 / 1000) *
(0.9842 / 1000))

```

```

    PM10_AutoTons = VMTAuto_I * NPVF_VMTAuto * PM10_Rate_Auto * (1 / 1000) * (0.9842 /
1000) + (AutoTrips_I * NPVF_VMTAuto * Per_Cold_Auto * PM10_Cold_Auto * (1 / 1000) *
(0.9842 / 1000))
    CO_FrTons = FrVMT_I * NPVF_FrVMT * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) +
(FrTrips_I * NPVF_FrVMT * Per_Cold_Truck * CO_Cold_Truck * (1 / 1000) * (0.9842 / 1000))
    VOC_FrTons = FrVMT_I * NPVF_FrVMT * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000)
    NOX_FrTons = FrVMT_I * NPVF_FrVMT * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) +
(FrTrips_I * NPVF_FrVMT * Per_Cold_Truck * NOX_Cold_Truck * (1 / 1000) * (0.9842 / 1000))
    PM10_FrTons = FrVMT_I * NPVF_FrVMT * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000)
+ (FrTrips_I * NPVF_FrVMT * Per_Cold_Truck * PM10_Cold_Truck * (1 / 1000) * (0.9842 / 1000))
    CO_RailTons = RailVMT_I * NPVF_RailVMT * CO_Rate_Rail * (1 / 1000) * (0.9842 / 1000)
    VOC_RailTons = RailVMT_I * NPVF_RailVMT * VOC_Rate_Rail * (1 / 1000) * (0.9842 /
1000)
    NOX_RailTons = RailVMT_I * NPVF_RailVMT * NOX_Rate_Rail * (1 / 1000) * (0.9842 /
1000)
    PM10_RailTons = RailVMT_I * NPVF_RailVMT * PM10_Rate_Rail * (1 / 1000) * (0.9842 /
1000)
    CO_BusTons = BusVMT_I * NPVF_BusVMT * CO_Rate_Bus * (1 / 1000) * (0.9842 / 1000)
    VOC_BusTons = BusVMT_I * NPVF_BusVMT * VOC_Rate_Bus * (1 / 1000) * (0.9842 /
1000)
    NOX_BusTons = BusVMT_I * NPVF_BusVMT * NOX_Rate_Bus * (1 / 1000) * (0.9842 /
1000)
    PM10_BusTons = BusVMT_I * NPVF_BusVMT * PM10_Rate_Bus * (1 / 1000) * (0.9842 /
1000)
    CO_Tons = CO_AutoTons + CO_FrTons + CO_RailTons + CO_BusTons
    VOC_Tons = VOC_AutoTons + VOC_FrTons + VOC_RailTons + VOC_BusTons
    NOX_Tons = NOX_AutoTons + NOX_FrTons + NOX_RailTons + NOX_BusTons
    PM10_Tons = PM10_AutoTons + PM10_FrTons + PM10_RailTons + PM10_BusTons
    'Set module level values (to be displayed)
    m_Values.Add m_CO_Tons, "CO_TONS"
    m_Values.Add m_VOC_Tons, "VOC_TONS"
    m_Values.Add m_NOX_Tons, "NOX_TONS"
    m_Values.Add m_PM10_Tons, "PM10_TONS"

    If IsNumeric(LogVMTAuto) And IsNumeric(N) And N > 0 Then
        NPVF_AutoEnvBen = (Exp(((LogVMTAuto / N) - Discount_Rate) * N) - 1) / ((LogVMTAuto
/ N) - Discount_Rate)
    Else
        NPVF_VMTAuto = Null
    End If
    VMTAuto_Tot = VMTAuto_I * NPVFAuto_VMT
    If IsNumeric(LogFrVMT) And IsNumeric(N) And N > 0 Then
        NPVF_FrEnvBen = (Exp(((LogFrVMT / N) - Discount_Rate) * N) - 1) / ((LogFrVMT / N) -
Discount_Rate)
    Else
        NPVF_FrVMT = Null
    End If
    FrVMT_Tot = FrVMT_I * NPVF_FrVMT
    If IsNumeric(LogBusVMT) And IsNumeric(N) And N > 0 Then
        NPVF_RailEnvBen = (Exp(((LogBusVMT / N) - Discount_Rate) * N) - 1) / ((LogBusVMT /
N) - Discount_Rate)
    Else
        NPVF_BusVMT = Null
    End If
    BusVMT_Tot = BusVMT_I * NPVF_BusVMT
    If IsNumeric(LogRailVMT) And IsNumeric(N) And N > 0 Then

```



```

    NPVF_BusEnvBen = (Exp(((LogRailVMT / N) - Discount_Rate) * N) - 1) / ((LogRailVMT /
N) - Discount_Rate)
    Else
    NPVF_RailVMT = Null
    End If

    CO_AutoBen = VMTAuto_I * CO_Rate_Auto * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_AutoEnvBen
    VOC_AutoBen = VMTAuto_I * VOC_Rate_Auto * (1 / 1000) * (0.9842 / 1000) *
VOCTon_Cost * NPVF_AutoEnvBen
    NOX_AutoBen = VMTAuto_I * NOX_Rate_Auto * (1 / 1000) * (0.9842 / 1000) *
NOXTon_Cost * NPVF_AutoEnvBen
    PM10_AutoBen = VMTAuto_I * PM10_Rate_Auto * (1 / 1000) * (0.9842 / 1000) *
PM10Ton_Cost * NPVF_AutoEnvBen
    CO_FrBen = FrVMT_I * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_FrEnvBen
    VOC_FrBen = FrVMT_I * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * VOCTon_Cost *
NPVF_FrEnvBen
    NOX_FrBen = FrVMT_I * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * NOXTon_Cost *
NPVF_FrEnvBen
    PM10_Fr = FrVMT_I * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * PM10Ton_Cost *
NPVF_FrEnvBen
    CO_RailBen = RailVMT_I * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_RailEnvBen
    VOC_RailBen = RailVMT_I * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000) *
VOCTon_Cost * NPVF_RailEnvBen
    NOX_RailBen = RailVMT_I * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) *
NOXTon_Cost * NPVF_RailEnvBen
    PM10_Rail = RailVMT_I * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * PM10Ton_Cost
* NPVF_RailEnvBen
    CO_BusBen = BusVMT_I * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_BusEnvBen
    VOC_BusBen = BusVMT_I * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000) *
VOCTon_Cost * NPVF_BusEnvBen
    NOX_BusBen = BusVMT_I * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) *
NOXTon_Cost * NPVF_BusEnvBen
    PM10_Bus = BusVMT_I * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * PM10Ton_Cost
* NPVF_BusEnvBen

    m_Env_Ben = CO_AutoBen + VOC_AutoBen + NOX_AutoBen + PM10_AutoBen +
CO_FrBen + VOC_FrBen +
    NOX_FrBen + PM10_FrBen + BusBen + VOC_BusBen + NOX_BusBen +
PM10_BusBen +
    CO_RailBen + VOC_RailBen + NOX_RailBen + PM10_RailBen
    'Set module level values (to be displayed)
    m_Values.Add m_Env_Ben, "ENV_BEN"

'Revenue Transfer
If IsNumeric(Revenue_F) And IsNumeric(Revenue_I) And IsNumeric(N) And N > 0 Then
    LogRevenue = Log((Revenue_F / Revenue_I))
    NPVF_Revenue = (Exp(((LogRevenue / N) - Discount_Rate) * N) - 1) / ((LogRevenue / N)
- Discount_Rate)
    Else
    NPVF_Revenue = Null
    End If
    m_User_Transfer = Revenue_I * NPVF_Revenue

```

```

'Set module level values (to be displayed)
  m_Values.Add m_Env_Ben, "USER_TRANSFER"

'SAFETY Calculations
'accident Calculations
  Dim hclass As Integer
  hclass = 2
  Fat_Rate_Auto = LookupFatalityRate(m_dbs, hclass, s_Fatality, veht_auto)
  If m_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(m_rst("Fat_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
  End If
  Inj_Rate_Auto = LookupFatalityRate(m_dbs, hclass, s_Injury, veht_auto)
  If m_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(m_rst("Inj_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
  End If
  Prop_Rate_Auto = LookupFatalityRate(m_dbs, hclass, s_Property, veht_auto)
  If m_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(m_rst("Prop_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
  End If
  Fat_Rate_Truck = LookupFatalityRate(m_dbs, hclass, s_Fatality, veht_truck)
  If m_rst("Fat_Rate_Auto") <> Fat_Rate_Truck Or IsNull(m_rst("Fat_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "Fat_Rate_Truck", Fat_Rate_Truck)
  End If
  Inj_Rate_Truck = LookupFatalityRate(m_dbs, hclass, s_Injury, veht_truck)
  If m_rst("Inj_Rate_Truck") <> Inj_Rate_Truck Or IsNull(m_rst("Inj_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "Inj_Rate_Truck", Inj_Rate_Truck)
  End If
  Prop_Rate_Truck = LookupFatalityRate(m_dbs, hclass, s_Property, veht_truck)
  If m_rst("Prop_Rate_Truck") <> Prop_Rate_Truck Or IsNull(m_rst("Prop_Rate_Truck"))
Then
    Call UpdateRstField(m_rst, "Prop_Rate_Truck", Prop_Rate_Truck)
  End If
  Fatality_Auto = VMTAuto_I * NPVF_VMTAuto * Fat_Rate_Auto / 100000000
  Injury_Auto = VMTAuto_I * NPVF_VMTAuto * Inj_Rate_Auto / 1000000
  Property_Auto = VMTAuto_I * NPVF_VMTAuto * Prop_Rate_Auto / 1000000
  Fatality_Fr = FrVMT_I * NPVF_FrVMT * Fat_Rate_Truck / 100000000
  Injury_Fr = FrVMT_I * NPVF_FrVMT * Inj_Rate_Truck / 1000000
  Property_Fr = FrVMT_I * NPVF_FrVMT * Prop_Rate_Truck / 1000000
  Fatality_Bus = VMTBus_I * NPVF_VMTBus * m_rst("Fat_Rate_Bus") / 100000000
  Injury_Bus = VMTBus_I * NPVF_VMTBus * m_rst("Inj_Rate_Bus") / 1000000
  Property_Bus = VMTBus_I * NPVF_VMTBus * m_rst("Prop_Rate_Bus") / 1000000
  Fatality_Rail = VMTRail_I * NPVF_VMTRail * m_rst("Fat_Rate_Rail") / 100000000
  Injury_Rail = VMTRail_I * NPVF_VMTRail * m_rst("Inj_Rate_Rail") / 1000000
  Property_Rail = VMTRail_I * NPVF_VMTRail * m_rst("Prop_Rate_Rail") / 1000000
  m_Fatality = Fatality_Auto + Fatality_Fr + Fatality_Rail + Fatality_Bus
  m_Injury = Injury_Auto + Injury_Fr + Injury_Rail + Injury_Bus
  m_Property = Property_Auto + Property_Fr + Property_Rail + Property_Bus
'Set module level values (to be displayed)
  m_Values.Add m_Fatality, "FATALITY"
  m_Values.Add m_Injury, "INJURY"
  m_Values.Add m_Property, "PROPERTY"

'Safety Benefit Calculations
  NPVF_AutoSafety = NPVF_AutoEnvBen
  Fatality_AutoBen = VMTAuto_I * Fat_Rate_Auto / 100000000 * Cost_Fatality *
NPVF_AutoSafety

```

```

    Injury_AutoBen = VMTAuto_I * Inj_Rate_Auto / 1000000 * Cost_Evident * NPVF_AutoSafety
    Prop_AutoBen = VMTAuto_I * Prop_Rate_Auto / 1000000 * Cost_PDO * NPVF_AutoSafety
    NPVF_FrSafety = NPVF_FrEnvBen
    Fatality_FrBen = VMTAuto_I * Fat_Rate_Truck / 100000000 * Cost_Fatality *
NPVF_FrSafety
    Injury_FrBen = VMTAuto_I * Inj_Rate_Truck / 1000000 * Cost_Evident * NPVF_FrSafety
    Prop_FrBen = VMTAuto_I * Prop_Rate_Truck / 1000000 * Cost_PDO * NPVF_FrSafety
    NPVF_BusSafety = NPVF_BusEnvBen
    Fatality_BusBen = VMTBus_I * NPVF_VMTBus * m_rst("Fat_Rate_Bus") / 100000000 *
Cost_Fatality * NPVF_BusSafety
    Injury_BusBen = VMTBus_I * NPVF_VMTBus * m_rst("Inj_Rate_Bus") / 1000000 *
Cost_Evident * NPVF_BusSafety
    Property_BusBen = VMTBus_I * NPVF_VMTBus * m_rst("Prop_Rate_Bus") / 1000000 *
Cost_PDO * NPVF_BusSafety
    NPVF_RailSafety = NPVF_RailEnvBen
    Fatality_RailBen = VMTRail_I * NPVF_VMTRail * m_rst("Fat_Rate_Rail") / 100000000 *
Cost_Fatality * NPVF_RailSafety
    Injury_RailBen = VMTRail_I * NPVF_VMTRail * m_rst("Inj_Rate_Rail") / 1000000 *
Cost_Evident * NPVF_RailSafety
    Property_RailBen = VMTRail_I * NPVF_VMTRail * m_rst("Prop_Rate_Rail") / 1000000 *
Cost_PDO * NPVF_RailSafety

    Safety_Ben = Fatality_AutoBen + Injury_AutoBen + Prop_AutoBen + Fatality_FrBen +
Injury_FrBen + _
                Prop_FrBen + Fatality_RailBen + Injury_RailBen + Prop_RailBen + _
                Fatality_BusBen + Injury_BusBen + Prop_BusBen
    'Set module level values (to be displayed)
    m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
tCost = Total_Cost + OpMaint_Cost - Terminal_Cost - EnvRetrofit_Cost
If tCost <> 0 Then
    m_BCR = (m_TT_Ben + m_User_Ben + m_Safety_Ben + m_Env_Ben) / tCost
End If
m_WSDOT_BCR = (m_TT_Ben + m_User_Ben + m_Safety_Ben + m_Env_Ben) /
(WSDOT_Cost + OpMaint - EnvRetrofit)

'Set module level values (to be displayed)
m_Values.Add m_Total_Cost, "TOTAL_COST"
m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

End Sub

Private Sub CalculateOutObjScores()

'Calculation for the System Operation and Maintenance
m_Sys_OM = (m_rst("Q1A") * 34) + (33 * m_rst("Q1B")) + (33 * m_rst("Q1D"))
m_Values.Add CInt(m_Sys_OM), "SYS_OM"
'Calculation for the System Preservation
m_Sys_Pres = 100 * m_rst("Q2A")
m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
m_Sp_Needs = 100 * m_rst("Q3A")

```

```

    m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = m_Values.item("TT_BEN")
  Else
    m_Cong_Rel = 0
  End If
  m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
  m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
  m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
  m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  ' This must be updated later
  m_Safety = m_Safety_Ben
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  m_Commnty = ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B"))) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E"))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") > 0 Then
    m_Collab = 50
  Else
    m_Collab = 0
  End If
  m_Collab = m_Collab + (50 * m_rst("Q10A"))
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
  ' This must be updated later
  m_Air_Qual = m_Env_Ben
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
    (33 * m_rst("Q15C"))
  m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
  m_Habitat = CInt(((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
    (25 * m_rst("Q16C")) + (25 * m_rst("Q16D")))) / _
    (1 + m_rst("Q16E") + m_rst("Q16F"))
  m_Values.Add m_Habitat, "HABITAT"

```

```
'Calculation for the Use of Resources
  m_Resource = 100 * m_rst("Q17")
  m_Values.Add m_Resource, "RESOURCE"
```

```
End Sub
```

```
Public Function GetItemValue(itemname As String) As Variant
  Dim rtnval
```

```
  rtnval = m_Values(itemname)
```

```
  GetItemValue = rtnval
```

```
End Function
```

```
Public Function DisplayFormat(fld As String, dval As Variant) As Variant
```

```
  Dim crncylist As String
```

```
  Dim intlist As String
```

```
  Dim dbllist As String
```

```
  'String to list the fields with currency formats
```

```
  crncylist = "TT_BEN,USER_BEN,USER_TRANSFER,ENV_BEN,SAFETY_BEN," & _
    "TOTAL_COST,WSDOT_COST,FEDERAL_COST"
```

```
  intlist = "TT_MIN,SYS_OM,SYS_PRES,SP_NEEDS,CONG_REL,TRAV_OPT,SEAMLESS," &
```

```
  "O_SAFETY,SECURITY,COMMNTY,COLLAB,FREIGHT,ECON_PROS," & _
    "TOURISM,AIR_QUAL,WTR_QUAL,HABITAT,RESOURCE"
```

```
  dbllist = "FATALITY,INJURY,PROPERTY,CO_TONS,VOC_TONS,NOX_TONS," & _
    "PM10_TONS,BCR,WSDOT_BCR"
```

```
  If InStr(crncylist, fld) <> 0 Then
```

```
    If dval <> 0 Then
```

```
      dval = Format(dval, "$#,###")
```

```
    Else
```

```
      dval = "$0"
```

```
    End If
```

```
  ElseIf InStr(intlist, fld) <> 0 Then
```

```
    dval = Round(dval, 0)
```

```
  ElseIf InStr(dbllist, fld) <> 0 Then
```

```
    dval = Round(dval, 2)
```

```
  End If
```

```
  DisplayFormat = dval
```

```
End Function
```

```
Private Function UpdateRstField(ByRef m_rst As DAO.Recordset, fld As String, fldval As Variant)
  As Boolean
```

```
  On Error Resume Next
```

```
  'Initailize return value
```

```
  UpdateRstField = True
```

```
  'Edit the fields value
```

```
  m_rst.Edit
```

```
    m_rst(fld) = fldval
```

```
  m_rst.Update
```

```
  'If there was an error clear it and return false to indicate that nothing occurred
```

```
  If Err <> 0 Then
```

```

    UpdateRstField = False
    Err.Clear
End If
End Function

```

```

Private Sub Class_Terminate()
    Set m_dbsMICA = Nothing
    Set m_rstfryOO = Nothing
    Set m_Values = Nothing
End Sub

```

SPASM Calculations

Option Compare Database

'Variables for the class level database objects

```

Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim m_Values As Collection

```

'Variables to hold the calculated values (variants)

```

Dim m_TT_Min
Dim m_TT_Ben
Dim m_User_Ben
Dim m_User_Transfer
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Terminal_Cost
Dim m_BCR
Dim m_WSDOT_BCR

```

'Variables to hold the outcome objective scores (variants)

```

Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Pro
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource

```

'Speed Assumed for lookup table

```
Private Const AUTOSPEED = 35
```

```
Public Function SetProjectNumber(pid As Long) As Boolean
```

```
    Dim rtnval As Boolean
    Dim qryDef As DAO.QueryDef

    'Initialize return value to false - set true if one record is returned
    rtnval = False

    Set m_dbs = CurrentDb

    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs("tran_calc_SPASM")
    qryDef.Parameters!projID = pid

    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If it is a unique data entry then calculate all scores
    If Not m_rst.EOF Then
        Set m_Values = New Collection
        CalculateBenefits
    End If

    ' This is done because of two many fields if all in one recordset
    'Open Second QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs("tran_calc_SPASM_outobj")
    qryDef.Parameters!projID = pid
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If it is a unique data entry then calculate all scores
    If Not m_rst.EOF Then
        CalculateOutObjScores
        rtnval = True
    End If

    SetProjectNumber = rtnval
End Function
```

```
Private Sub CalculateBenefits()
```

```
    'In Vehicle Travel Time Calculations:
    Ann_Daily_Benefit = m_rst("Ann_Daily_Benefit")
    PP_InVehAuto_PCI = m_rst("PP_InVehAuto_PCI")
    PP_InVehAuto_BCI = m_rst("PP_InVehAuto_BCI")
    PP_InVehAuto_PCF = m_rst("PP_InVehAuto_PCF")
    PP_InVehAuto_BCF = m_rst("PP_InVehAuto_BCF")
    PP_InVehFr_PCI = m_rst("PP_InVehFr_PCI")
    PP_InVehFr_BCI = m_rst("PP_InVehFr_BCI")
    PP_InVehFr_PCF = m_rst("PP_InVehFr_PCF")
    PP_InVehFr_BCF = m_rst("PP_InVehFr_BCF")
    PP_InVehBus_PCI = m_rst("PP_InVehBus_PCI")
    PP_InVehBus_BCI = m_rst("PP_InVehBus_BCI")
    PP_InVehBus_PCF = m_rst("PP_InVehBus_PCF")
    PP_InVehBus_BCF = m_rst("PP_InVehBus_BCF")
    PP_InVehRail_PCI = m_rst("PP_InVehRail_PCI")
    PP_InVehRail_BCI = m_rst("PP_InVehRail_BCI")
    PP_InVehRail_PCF = m_rst("PP_InVehRail_PCF")
    PP_InVehRail_BCF = m_rst("PP_InVehRail_BCF")
    OP_InVehAuto_PCI = m_rst("OP_InVehAuto_PCI")
```

```

OP_InVehAuto_BCI = m_rst("OP_InVehAuto_BCI")
OP_InVehAuto_PCF = m_rst("OP_InVehAuto_PCF")
OP_InVehAuto_BCF = m_rst("OP_InVehAuto_BCF")
OP_InVehFr_PCI = m_rst("OP_InVehFr_PCI")
OP_InVehFr_BCI = m_rst("OP_InVehFr_BCI")
OP_InVehFr_PCF = m_rst("OP_InVehFr_PCF")
OP_InVehFr_BCF = m_rst("OP_InVehFr_BCF")
OP_InVehBus_PCI = m_rst("OP_InVehBus_PCI")
OP_InVehBus_BCI = m_rst("OP_InVehBus_BCI")
OP_InVehBus_PCF = m_rst("OP_InVehBus_PCF")
OP_InVehBus_BCF = m_rst("OP_InVehBus_BCF")
OP_InVehRail_PCI = m_rst("OP_InVehRail_PCI")
OP_InVehRail_BCI = m_rst("OP_InVehRail_BCI")
OP_InVehRail_PCF = m_rst("OP_InVehRail_PCF")
OP_InVehRail_BCF = m_rst("OP_InVehRail_BCF")

```

```

PP_OutVehAuto_PCI = m_rst("PP_OutVehAuto_PCI")
PP_OutVehAuto_BCI = m_rst("PP_OutVehAuto_BCI")
PP_OutVehAuto_PCF = m_rst("PP_OutVehAuto_PCF")
PP_OutVehAuto_BCF = m_rst("PP_OutVehAuto_BCF")
PP_OutVehFr_PCI = m_rst("PP_OutVehFr_PCI")
PP_OutVehFr_BCI = m_rst("PP_OutVehFr_BCI")
PP_OutVehFr_PCF = m_rst("PP_OutVehFr_PCF")
PP_OutVehFr_BCF = m_rst("PP_OutVehFr_BCF")
PP_OutVehBus_PCI = m_rst("PP_OutVehBus_PCI")
PP_OutVehBus_BCI = m_rst("PP_OutVehBus_BCI")
PP_OutVehBus_PCF = m_rst("PP_OutVehBus_PCF")
PP_OutVehBus_BCF = m_rst("PP_OutVehBus_BCF")
PP_OutVehRail_PCI = m_rst("PP_OutVehRail_PCI")
PP_OutVehRail_BCI = m_rst("PP_OutVehRail_BCI")
PP_OutVehRail_PCF = m_rst("PP_OutVehRail_PCF")
PP_OutVehRail_BCF = m_rst("PP_OutVehRail_BCF")
OP_OutVehAuto_PCI = m_rst("OP_OutVehAuto_PCI")
OP_OutVehAuto_BCI = m_rst("OP_OutVehAuto_BCI")
OP_OutVehAuto_PCF = m_rst("OP_OutVehAuto_PCF")
OP_OutVehAuto_BCF = m_rst("OP_OutVehAuto_BCF")
OP_OutVehFr_PCI = m_rst("OP_OutVehFr_PCI")
OP_OutVehFr_BCI = m_rst("OP_OutVehFr_BCI")
OP_OutVehFr_PCF = m_rst("OP_OutVehFr_PCF")
OP_OutVehFr_BCF = m_rst("OP_OutVehFr_BCF")
OP_OutVehBus_PCI = m_rst("OP_OutVehBus_PCI")
OP_OutVehBus_BCI = m_rst("OP_OutVehBus_BCI")
OP_OutVehBus_PCF = m_rst("OP_OutVehBus_PCF")
OP_OutVehBus_BCF = m_rst("OP_OutVehBus_BCF")
OP_OutVehRail_PCI = m_rst("OP_OutVehRail_PCI")
OP_OutVehRail_BCI = m_rst("OP_OutVehRail_BCI")
OP_OutVehRail_PCF = m_rst("OP_OutVehRail_PCF")
OP_OutVehRail_BCF = m_rst("OP_OutVehRail_BCF")

```

'Yearly Travel Time Benefits in Minutes

```

InVehAuto_I = ((PP_InVehAuto_PCI - PP_InVehAuto_BCI) + (OP_InVehAuto_PCI -
OP_InVehAuto_BCI)) * Ann_Daily_Benefit
InVehFr_I = ((PP_InVehFr_PCI - PP_InVehFr_BCI) + (OP_InVehFr_PCI - OP_InVehFr_BCI)) *
Ann_Daily_Benefit
InVehBus_I = ((PP_InVehBus_PCI - PP_InVehBus_BCI) + (OP_InVehBus_PCI - OP_InVehBus_BCI))
* Ann_Daily_Benefit
InVehRail_I = ((PP_InVehRail_PCI - PP_InVehRail_BCI) + (OP_InVehRail_PCI - OP_InVehRail_BCI))
* Ann_Daily_Benefit
InVehAuto_F = ((PP_InVehAuto_PCF - PP_InVehAuto_BCF) + (OP_InVehAuto_PCF -
OP_InVehAuto_BCF)) * Ann_Daily_Benefit

```



```

    InVehFr_F = ((PP_InVehFr_PCF - PP_InVehFr_BCF) + (OP_InVehFr_PCF - OP_InVehFr_BCF)) *
Ann_Daily_Benefit
    InVehBus_F = ((PP_InVehBus_PCF - PP_InVehBus_BCF) + (OP_InVehBus_PCF -
OP_InVehBus_BCF)) * Ann_Daily_Benefit
    InVehRail_F = ((PP_InVehRail_PCF - PP_InVehRail_BCF) + (OP_InVehRail_PCF -
OP_InVehRail_BCF)) * Ann_Daily_Benefit
    'Time value the same for auto, bus, and rail travel
    InVeh_I = InVehAuto_I + InVehBus_I + InVehRail_I
    InVeh_F = InVehAuto_F + InVehBus_F + InVehRail_F
    'Out of Vehicle Travel Time Calculations:
    'Yearly Travel Time Benefits in Minutes
    OutVehAuto_I = ((PP_OutVehAuto_PCI - PP_OutVehAuto_BCI) + (OP_OutVehAuto_PCI -
OP_OutVehAuto_BCI)) * Ann_Daily_Benefit
    OutVehFr_I = ((PP_OutVehFr_PCI - PP_OutVehFr_BCI) + (OP_OutVehFr_PCI - OP_OutVehFr_BCI))
* Ann_Daily_Benefit
    OutVehBus_I = ((PP_OutVehBus_PCI - PP_OutVehBus_BCI) + (OP_OutVehBus_PCI -
OP_OutVehBus_BCI)) * Ann_Daily_Benefit
    OutVehRail_I = ((PP_OutVehRail_PCI - PP_OutVehRail_BCI) + (OP_OutVehRail_PCI -
OP_OutVehRail_BCI)) * Ann_Daily_Benefit
    OutVehAuto_F = ((PP_OutVehAuto_PCF - PP_OutVehAuto_BCF) + (OP_OutVehAuto_PCF -
OP_OutVehAuto_BCF)) * Ann_Daily_Benefit
    OutVehFr_F = ((PP_OutVehFr_PCF - PP_OutVehFr_BCF) + (OP_OutVehFr_PCF -
OP_OutVehFr_BCF)) * Ann_Daily_Benefit
    OutVehBus_F = ((PP_OutVehBus_PCF - PP_OutVehBus_BCF) + (OP_OutVehBus_PCF -
OP_OutVehBus_BCF)) * Ann_Daily_Benefit
    OutVehRail_F = ((PP_OutVehRail_PCF - PP_OutVehRail_BCF) + (OP_OutVehRail_PCF -
OP_OutVehRail_BCF)) * Ann_Daily_Benefit
    'Time value the same for auto, bus, and rail travel
    OutVeh_I = OutVehAuto_I + OutVehBus_I + OutVehRail_I
    OutVeh_F = OutVehAuto_F + OutVehBus_F + OutVehRail_F
    TT_Min_I = InVeh_I + InVehFr_I + OutVeh_I + OutVehFr_I
    TT_Min_F = InVeh_F + InVehFr_F + OutVeh_F + OutVehFr_F
    FrTT_Min_I = InVehFr_I + OutVehFr_I
    FrTT_Min_F = InVehFr_F + OutVehFr_F

'Induced Ridership
N = m_rst("Fore_Year") - m_rst("Init_Year") + 1
If IsNumeric(TT_Min_F) And IsNumeric(TT_Min_I) And IsNumeric(N) And N > 0 Then
    LogTT_Min = Log((TT_Min_F / TT_Min_I))
    NPVF_Min = (Exp(((LogTT_Min / N)) * N) - 1) / _
                ((LogTT_Min / N))
Else
    NPVF_Min = Null
End If
If IsNumeric(FrTT_Min_F) And IsNumeric(FrTT_Min_I) And IsNumeric(N) And N > 0 Then
    LogTT_FrMin = Log((FrTT_Min_F / FrTT_Min_I))
    NPVF_FrMin = (Exp(((LogTT_FrMin / N)) * N) - 1) / _
                ((LogTT_FrMin / N))
Else
    NPVF_FrMin = Null
End If
m_TT_Min = TT_Min_I * NPVF_Min
FrTT_Min = FrTT_Min_I * NPVF_FrMin
'Set module level values (to be displayed)
m_Values.Add m_TT_Min, "TT_MIN"
m_Values.Add FrTT_Min, "FrTT_MIN"

'Travel Time Benefits
PP_TripsAuto_PCI = m_rst("PP_TripsAuto_PCI")
PP_TripsAuto_BCI = m_rst("PP_TripsAuto_BCI")
PP_TripsAuto_PCF = m_rst("PP_TripsAuto_PCF")
PP_TripsAuto_BCF = m_rst("PP_TripsAuto_BCF")

```

```

PP_TripsFr_PCI = m_rst("PP_TripsFr_PCI")
PP_TripsFr_BCI = m_rst("PP_TripsFr_BCI")
PP_TripsFr_PCF = m_rst("PP_TripsFr_PCF")
PP_TripsFr_BCF = m_rst("PP_TripsFr_BCF")
PP_TripsBus_PCI = m_rst("PP_TripsBus_PCI")
PP_TripsBus_BCI = m_rst("PP_TripsBus_BCI")
PP_TripsBus_PCF = m_rst("PP_TripsBus_PCF")
PP_TripsBus_BCF = m_rst("PP_TripsBus_BCF")
PP_TripsRail_PCI = m_rst("PP_TripsRail_PCI")
PP_TripsRail_BCI = m_rst("PP_TripsRail_BCI")
PP_TripsRail_PCF = m_rst("PP_TripsRail_PCF")
PP_TripsRail_BCF = m_rst("PP_TripsRail_BCF")
OP_TripsAuto_PCI = m_rst("OP_TripsAuto_PCI")
OP_TripsAuto_BCI = m_rst("OP_TripsAuto_BCI")
OP_TripsAuto_PCF = m_rst("OP_TripsAuto_PCF")
OP_TripsAuto_BCF = m_rst("OP_TripsAuto_BCF")
OP_TripsFr_PCI = m_rst("OP_TripsFr_PCI")
OP_TripsFr_BCI = m_rst("OP_TripsFr_BCI")
OP_TripsFr_PCF = m_rst("OP_TripsFr_PCF")
OP_TripsFr_BCF = m_rst("OP_TripsFr_BCF")
OP_TripsBus_PCI = m_rst("OP_TripsBus_PCI")
OP_TripsBus_BCI = m_rst("OP_TripsBus_BCI")
OP_TripsBus_PCF = m_rst("OP_TripsBus_PCF")
OP_TripsBus_BCF = m_rst("OP_TripsBus_BCF")
OP_TripsRail_PCI = m_rst("OP_TripsRail_PCI")
OP_TripsRail_BCI = m_rst("OP_TripsRail_BCI")
OP_TripsRail_PCF = m_rst("OP_TripsRail_PCF")
OP_TripsRail_BCF = m_rst("OP_TripsRail_BCF")

TripsAuto_I = ((PP_TripsAuto_PCI - PP_TripsAuto_BCI) + (OP_TripsAuto_PCI - OP_TripsAuto_BCI)) *
Ann_Daily_Benefit
TripsFr_I = ((PP_TripsFr_PCI - PP_TripsFr_BCI) + (OP_TripsFr_PCI - OP_TripsFr_BCI)) *
Ann_Daily_Benefit
TripsBus_I = ((PP_TripsBus_PCI - PP_TripsBus_BCI) + (OP_TripsBus_PCI - OP_TripsBus_BCI)) *
Ann_Daily_Benefit
TripsRail_I = ((PP_TripsRail_PCI - PP_TripsRail_BCI) + (OP_TripsRail_PCI - OP_TripsRail_BCI)) *
Ann_Daily_Benefit
TripsAuto_F = ((PP_TripsAuto_PCF - PP_TripsAuto_BCF) + (OP_TripsAuto_PCF -
OP_TripsAuto_BCF)) * Ann_Daily_Benefit
TripsFr_F = ((PP_TripsFr_PCF - PP_TripsFr_BCF) + (OP_TripsFr_PCF - OP_TripsFr_BCF)) *
Ann_Daily_Benefit
TripsBus_F = ((PP_TripsBus_PCF - PP_TripsBus_BCF) + (OP_TripsBus_PCF - OP_TripsBus_BCF)) *
Ann_Daily_Benefit
TripsRail_F = ((PP_TripsRail_PCF - PP_TripsRail_BCF) + (OP_TripsRail_PCF - OP_TripsRail_BCF)) *
Ann_Daily_Benefit
'Travel Time Benefits in Dollars
Percent_TV_InVeh = m_rst("Percent_TV_InVeh")
Time_Value_Veh = m_rst("Time_Value_Veh")
Time_Value_Freight = m_rst("Time_Value_Truck")
Discount_Rate = m_rst("Discount_Rate")
TTAuto_Ben_I = ((InVeh_I) / 60 * (Percent_TV_InVeh * Time_Value_Veh)) + (OutVeh_I) / 60 *
(Percent_TV_OutVeh * Time_Value_Veh)
TTAuto_Ben_F = ((InVeh_F) / 60 * (Percent_TV_InVeh * Time_Value_Veh)) + (OutVeh_F) / 60 *
(Percent_TV_OutVeh * Time_Value_Veh)
FrTT_Ben_I = ((InVehFr_I) / 60 * (Percent_TV_InVeh * Time_Value_Freight)) + (OutVehFr_I) / 60 *
(Percent_TV_OutVeh * Time_Value_Freight)
FrTT_Ben_F = ((InVehFr_F) / 60 * (Percent_TV_InVeh * Time_Value_Freight)) + (OutVehFr_F) / 60 *
(Percent_TV_OutVeh * Time_Value_Freight)
TT_Ben_I = TTAuto_Ben_I + FrTT_Ben_I
TT_Ben_F = TTAuto_Ben_F + FrTT_Ben_F
If IsNumeric(TT_Ben_F) And IsNumeric(TT_Ben_I) And IsNumeric(N) And N > 0 Then
LogTT_Ben = Log((TT_Ben_F / TT_Ben_I))

```

```

NPVF_TTBen = (Exp(((LogTT_Ben / N) - Discount_Rate) * N) - 1) / _
              ((LogTT_Ben / N) - Discount_Rate)
Else
  NPVF_TTBen = Null
End If
If IsNumeric(FrTT_Ben_F) And IsNumeric(FrTT_Ben_I) And IsNumeric(N) And N > 0 Then
  LogFrTT_Ben = Log((FrTT_Ben_F / FrTT_Ben_I))
  NPVF_FrTTBen = (Exp(((LogFrTT_Ben / N) - Discount_Rate) * N) - 1) / _
                 ((LogFrTT_Ben / N) - Discount_Rate)
Else
  NPVF_TTBen = Null
End If
m_TT_Ben = TT_Ben_I * NPVF_TTBen
FrTT_Ben = FrTT_Ben_I * NPVF_FrTTBen
'Set module level values (to be displayed)
  m_Values.Add m_TT_Ben, "TT_BEN"
  m_Values.Add FrTT_Ben, "FrTT_BEN"

'Operating Cost Calculations
PP_VMTAuto_PCI = m_rst("PP_VMTAuto_PCI")
PP_VMTAuto_BCI = m_rst("PP_VMTAuto_BCI")
PP_VMTAuto_PCF = m_rst("PP_VMTAuto_PCF")
PP_VMTAuto_BCF = m_rst("PP_VMTAuto_BCF")
PP_VMTFr_PCI = m_rst("PP_VMTFr_PCI")
PP_VMTFr_BCI = m_rst("PP_VMTFr_BCI")
PP_VMTFr_PCF = m_rst("PP_VMTFr_PCF")
PP_VMTFr_BCF = m_rst("PP_VMTFr_BCF")
PP_VMTBus_PCI = m_rst("PP_VMTBus_PCI")
PP_VMTBus_BCI = m_rst("PP_VMTBus_BCI")
PP_VMTBus_PCF = m_rst("PP_VMTBus_PCF")
PP_VMTBus_BCF = m_rst("PP_VMTBus_BCF")
PP_VMTRail_PCI = m_rst("PP_VMTRail_PCI")
PP_VMTRail_BCI = m_rst("PP_VMTRail_BCI")
PP_VMTRail_PCF = m_rst("PP_VMTRail_PCF")
PP_VMTRail_BCF = m_rst("PP_VMTRail_BCF")
OP_VMTAuto_PCI = m_rst("OP_VMTAuto_PCI")
OP_VMTAuto_BCI = m_rst("OP_VMTAuto_BCI")
OP_VMTAuto_PCF = m_rst("OP_VMTAuto_PCF")
OP_VMTAuto_BCF = m_rst("OP_VMTAuto_BCF")
OP_VMTFr_PCI = m_rst("OP_VMTFr_PCI")
OP_VMTFr_BCI = m_rst("OP_VMTFr_BCI")
OP_VMTFr_PCF = m_rst("OP_VMTFr_PCF")
OP_VMTFr_BCF = m_rst("OP_VMTFr_BCF")
OP_VMTBus_PCI = m_rst("OP_VMTBus_PCI")
OP_VMTBus_BCI = m_rst("OP_VMTBus_BCI")
OP_VMTBus_PCF = m_rst("OP_VMTBus_PCF")
OP_VMTBus_BCF = m_rst("OP_VMTBus_BCF")
OP_VMTRail_PCI = m_rst("OP_VMTRail_PCI")
OP_VMTRail_BCI = m_rst("OP_VMTRail_BCI")
OP_VMTRail_PCF = m_rst("OP_VMTRail_PCF")
OP_VMTRail_BCF = m_rst("OP_VMTRail_BCF")

VMTAuto_I = ((PP_VMTAuto_PCI - PP_VMTAuto_BCI) + (OP_VMTAuto_PCI - OP_VMTAuto_BCI)) *
Ann_Daily_Benefit
FrVMT_I = ((PP_VMTFr_PCI - PP_VMTFr_BCI) + (OP_VMTFr_PCI - OP_VMTFr_BCI)) *
Ann_Daily_Benefit
BusVMT_I = ((PP_VMTBus_PCI - PP_VMTBus_BCI) + (OP_VMTBus_PCI - OP_VMTBus_BCI)) *
Ann_Daily_Benefit
RailVMT_I = ((PP_VMTRail_PCI - PP_VMTRail_BCI) + (OP_VMTRail_PCI - OP_VMTRail_BCI)) *
Ann_Daily_Benefit
VMTAuto_F = ((PP_VMTAuto_PCF - PP_VMTAuto_BCF) + (OP_VMTAuto_PCF -
OP_VMTAuto_BCF)) * Ann_Daily_Benefit

```

```

    FrVMT_F = ((PP_VMTFr_PCF - PP_VMTFr_BCF) + (OP_VMTFr_PCF - OP_VMTFr_BCF)) *
Ann_Daily_Benefit
    BusVMT_F = ((PP_VMTBus_PCF - PP_VMTBus_BCF) + (OP_VMTBus_PCF - OP_VMTBus_BCF)) *
Ann_Daily_Benefit
    RailVMT_F = ((PP_VMTRail_PCF - PP_VMTRail_BCF) + (OP_VMTRail_PCF - OP_VMTRail_BCF)) *
Ann_Daily_Benefit

```

```

If IsNumeric(VMTAuto_I) And IsNumeric(VMTAuto_F) And IsNumeric(N) And N > 0 Then
    LogVMTAuto = Log((VMTAuto_F / VMTAuto_I))
    NPVF_VMTAuto = ((Exp(LogVMTAuto) - 1) / (LogVMTAuto) / N)
Else
    NPVF_VMTAuto = Null
End If
VMTAuto_Tot = VMTAuto_I * NPVFAuto_VMT
If IsNumeric(FrVMT_F) And IsNumeric(FrVMT_I) And IsNumeric(N) And N > 0 Then
    LogFrVMT = Log((FrVMT_F / FrVMT_I))
    NPVF_FrVMT = ((Exp(LogFrVMT) - 1) / (LogFrVMT) / N)
Else
    NPVF_FrVMT = Null
End If
FrVMT_Tot = FrVMT_I * NPVF_FrVMT
If IsNumeric(BusVMT_F) And IsNumeric(BusVMT_I) And IsNumeric(N) And N > 0 Then
    LogBusVMT = Log((BusVMT_F / BusVMT_I))
    NPVF_BusVMT = ((Exp(LogBusVMT) - 1) / (LogBusVMT) / N)
Else
    NPVF_BusVMT = Null
End If
BusVMT_Tot = BusVMT_I * NPVF_BusVMT
If IsNumeric(RailVMT_F) And IsNumeric(RailVMT_I) And IsNumeric(N) And N > 0 Then
    LogRailVMT = Log((RailVMT_F / RailVMT_I))
    NPVF_RailVMT = ((Exp(LogRailVMT) - 1) / (LogRailVMT) / N)
Else
    NPVF_RailVMT = Null
End If

```

'User benefit cost calculations

```

Veh_OpCost_Full = m_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = m_rst("Veh_OpCost_Direct")
Freight_OpCost = m_rst("Freight_OpCost")
If m_rst("Full_Cost") Then
    UserBen_I = (VMTAuto_I * Veh_OpCost_Full) + (FrVMT_I * Freight_OpCost)
    UserBen_F = (VMTAuto_F * Veh_OpCost_Full) + (FrVMT_F * Freight_OpCost)
Else
    UserBen_I = (VMTAuto_I * Veh_OpCost_Direct) + (FrVMT_I * Freight_OpCost)
    UserBen_F = (VMTAuto_F * Veh_OpCost_Direct) + (FrVMT_F * Freight_OpCost)
End If
If IsNumeric(UserBen_F) And IsNumeric(UserBen_I) And IsNumeric(N) And N > 0 Then
    LogUserBen = Log((UserBen_F / UserBen_I))
    NPVF_UCBen = (Exp(((LogUserBen / N) - Discount_Rate) * N) - 1) / ((LogUserBen / N) -
Discount_Rate)
Else
    NPVF_UCBen = Null
End If
m_User_Ben = UserBen_I * NPVF_UCBen
'Set module level values (to be displayed)
m_Values.Add m_User_Ben, "USER_BEN"

```

'Air Pollution

'Emissions Calculations

```

Per_Cold_Auto = m_rst("Per_Cold_Auto")
Per_Cold_Truck = m_rst("Per_Cold_Truck")
CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)

```

```

If m_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(m_rst("CO_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If
CO_Rate_Truck = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_truck)
If m_rst("CO_Rate_Truck") <> NOX_Rate_Truck Or IsNull(m_rst("CO_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "CO_Rate_Truck", CO_Rate_Truck)
End If
COTon_Cost = m_rst("COTon_Cost")
VOC_Rate_Auto = m_rst("VOC_Rate_Auto")
VOC_Rate_Truck = m_rst("VOC_Rate_Truck")
VOCTon_Cost = m_rst("VOCTon_Cost")
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
If m_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(m_rst("NOX_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
NOX_Rate_Truck = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_truck)
If m_rst("NOX_Rate_Truck") <> NOX_Rate_Truck Or IsNull(m_rst("NOX_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "NOX_Rate_Truck", NOX_Rate_Truck)
End If
NOXTon_Cost = m_rst("NOXTon_Cost")
PM10_Rate_Auto = m_rst("PM10_Rate_Auto")
PM10_Rate_Truck = m_rst("PM10_Rate_Truck")
PM10Ton_Cost = m_rst("PM10Ton_Cost")
CO_Rate_Rail = m_rst("CO_Rate_Rail")
VOC_Rate_Rail = m_rst("VOC_Rate_Rail")
NOX_Rate_Rail = m_rst("NOX_Rate_Rail")
PM10_Rate_Rail = m_rst("PM10_Rate_Rail")
CO_Rate_Bus = m_rst("CO_Rate_Bus")
VOC_Rate_Bus = m_rst("VOC_Rate_Bus")
NOX_Rate_Bus = m_rst("NOX_Rate_Bus")
PM10_Rate_Bus = m_rst("PM10_Rate_Bus")

CO_AutoTons = (VMTAuto_I * NPVF_VMTAuto * CO_Rate_Auto * (1 / 1000) * (0.9842 / 1000)) +
(AutoTrips_I * NPVF_VMTAuto * Per_Cold_Auto * CO_Cold_Auto * (1 / 1000) * (0.9842 / 1000))
VOC_AutoTons = (VMTAuto_I * NPVF_VMTAuto * VOC_Rate_Auto * (1 / 1000) * (0.9842 / 1000))
NOX_AutoTons = (VMTAuto_I * NPVF_VMTAuto * NOX_Rate_Auto * (1 / 1000) * (0.9842 / 1000)) +
(AutoTrips_I * NPVF_VMTAuto * Per_Cold_Auto * VOC_Cold_Auto * (1 / 1000) * (0.9842 / 1000))
PM10_AutoTons = VMTAuto_I * NPVF_VMTAuto * PM10_Rate_Auto * (1 / 1000) * (0.9842 / 1000) +
(AutoTrips_I * NPVF_VMTAuto * Per_Cold_Auto * PM10_Cold_Auto * (1 / 1000) * (0.9842 / 1000))
CO_FrTons = FrVMT_I * NPVF_FrVMT * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) + (FrTrips_I *
NPVF_FrVMT * Per_Cold_Truck * CO_Cold_Truck * (1 / 1000) * (0.9842 / 1000))
VOC_FrTons = FrVMT_I * NPVF_FrVMT * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000)
NOX_FrTons = FrVMT_I * NPVF_FrVMT * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) + (FrTrips_I
* NPVF_FrVMT * Per_Cold_Truck * NOX_Cold_Truck * (1 / 1000) * (0.9842 / 1000))
PM10_FrTons = FrVMT_I * NPVF_FrVMT * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000) +
(FrTrips_I * NPVF_FrVMT * Per_Cold_Truck * PM10_Cold_Truck * (1 / 1000) * (0.9842 / 1000))
CO_RailTons = RailVMT_I * NPVF_RailVMT * CO_Rate_Rail * (1 / 1000) * (0.9842 / 1000)
VOC_RailTons = RailVMT_I * NPVF_RailVMT * VOC_Rate_Rail * (1 / 1000) * (0.9842 / 1000)
NOX_RailTons = RailVMT_I * NPVF_RailVMT * NOX_Rate_Rail * (1 / 1000) * (0.9842 / 1000)
PM10_RailTons = RailVMT_I * NPVF_RailVMT * PM10_Rate_Rail * (1 / 1000) * (0.9842 / 1000)
CO_BusTons = BusVMT_I * NPVF_BusVMT * CO_Rate_Bus * (1 / 1000) * (0.9842 / 1000)
VOC_BusTons = BusVMT_I * NPVF_BusVMT * VOC_Rate_Bus * (1 / 1000) * (0.9842 / 1000)
NOX_BusTons = BusVMT_I * NPVF_BusVMT * NOX_Rate_Bus * (1 / 1000) * (0.9842 / 1000)
PM10_BusTons = BusVMT_I * NPVF_BusVMT * PM10_Rate_Bus * (1 / 1000) * (0.9842 / 1000)
CO_Tons = CO_AutoTons + CO_FrTons + CO_RailTons + CO_BusTons
VOC_Tons = VOC_AutoTons + VOC_FrTons + VOC_RailTons + VOC_BusTons
NOX_Tons = NOX_AutoTons + NOX_FrTons + NOX_RailTons + NOX_BusTons
PM10_Tons = PM10_AutoTons + PM10_FrTons + PM10_RailTons + PM10_BusTons
'Set module level values (to be displayed)
m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"

```

```

m_Values.Add m_PM10_Tons, "PM10_TONS"

If IsNumeric(LogVMTAuto) And IsNumeric(N) And N > 0 Then
  NPVF_AutoEnvBen = (Exp(((LogVMTAuto / N) - Discount_Rate) * N) - 1) / ((LogVMTAuto / N) -
Discount_Rate)
Else
  NPVF_VMTAuto = Null
End If
VMTAuto_Tot = VMTAuto_I * NPVFAuto_VMT
If IsNumeric(LogFrVMT) And IsNumeric(N) And N > 0 Then
  NPVF_FrEnvBen = (Exp(((LogFrVMT / N) - Discount_Rate) * N) - 1) / ((LogFrVMT / N) -
Discount_Rate)
Else
  NPVF_FrVMT = Null
End If
FrVMT_Tot = FrVMT_I * NPVF_FrVMT
If IsNumeric(LogBusVMT) And IsNumeric(N) And N > 0 Then
  NPVF_RailEnvBen = (Exp(((LogBusVMT / N) - Discount_Rate) * N) - 1) / ((LogBusVMT / N) -
Discount_Rate)
Else
  NPVF_BusVMT = Null
End If
BusVMT_Tot = BusVMT_I * NPVF_BusVMT
If IsNumeric(LogRailVMT) And IsNumeric(N) And N > 0 Then
  NPVF_BusEnvBen = (Exp(((LogRailVMT / N) - Discount_Rate) * N) - 1) / ((LogRailVMT / N) -
Discount_Rate)
Else
  NPVF_RailVMT = Null
End If

CO_AutoBen = VMTAuto_I * CO_Rate_Auto * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_AutoEnvBen
VOC_AutoBen = VMTAuto_I * VOC_Rate_Auto * (1 / 1000) * (0.9842 / 1000) * VOCTon_Cost *
NPVF_AutoEnvBen
NOX_AutoBen = VMTAuto_I * NOX_Rate_Auto * (1 / 1000) * (0.9842 / 1000) * NOXTon_Cost *
NPVF_AutoEnvBen
PM10_AutoBen = VMTAuto_I * PM10_Rate_Auto * (1 / 1000) * (0.9842 / 1000) * PM10Ton_Cost *
NPVF_AutoEnvBen
CO_FrBen = FrVMT_I * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_FrEnvBen
VOC_FrBen = FrVMT_I * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * VOCTon_Cost *
NPVF_FrEnvBen
NOX_FrBen = FrVMT_I * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * NOXTon_Cost *
NPVF_FrEnvBen
PM10_Fr = FrVMT_I * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * PM10Ton_Cost *
NPVF_FrEnvBen
CO_RailBen = RailVMT_I * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_RailEnvBen
VOC_RailBen = RailVMT_I * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * VOCTon_Cost *
NPVF_RailEnvBen
NOX_RailBen = RailVMT_I * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * NOXTon_Cost *
NPVF_RailEnvBen
PM10_Rail = RailVMT_I * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * PM10Ton_Cost *
NPVF_RailEnvBen
CO_BusBen = BusVMT_I * CO_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * COTon_Cost *
NPVF_BusEnvBen
VOC_BusBen = BusVMT_I * VOC_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * VOCTon_Cost *
NPVF_BusEnvBen
NOX_BusBen = BusVMT_I * NOX_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * NOXTon_Cost *
NPVF_BusEnvBen
PM10_Bus = BusVMT_I * PM10_Rate_Truck * (1 / 1000) * (0.9842 / 1000) * PM10Ton_Cost *
NPVF_BusEnvBen

```

```

m_Env_Ben = CO_AutoBen + VOC_AutoBen + NOX_AutoBen + PM10_AutoBen + CO_FrBen +
VOC_FrBen +
NOX_FrBen + PM10_FrBen + BusBen + VOC_BusBen + NOX_BusBen + PM10_BusBen +
CO_RailBen + VOC_RailBen + NOX_RailBen + PM10_RailBen
'Set module level values (to be displayed)
m_Values.Add m_Env_Ben, "ENV_BEN"

```

```

'Revenue Transfer
If IsNumeric(Revenue_F) And IsNumeric(Revenue_I) And IsNumeric(N) And N > 0 Then
    LogRevenue = Log((Revenue_F / Revenue_I))
    NPVF_Revenue = (Exp(((LogRevenue / N) - Discount_Rate) * N) - 1) / ((LogRevenue / N) -
Discount_Rate)
Else
    NPVF_Revenue = Null
End If
m_User_Transfer = Revenue_I * NPVF_Revenue
'Set module level values (to be displayed)
m_Values.Add m_Env_Ben, "USER_TRANSFER"

```

'SAFETY Calculations

'accident Calculations

```

Dim hclass As Integer
hclass = 2
Fat_Rate_Auto = LookupFatalityRate(m_dbs, hclass, s_Fatality, veht_auto)
If m_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(m_rst("Fat_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
End If
Inj_Rate_Auto = LookupFatalityRate(m_dbs, hclass, s_Injury, veht_auto)
If m_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(m_rst("Inj_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
End If
Prop_Rate_Auto = LookupFatalityRate(m_dbs, hclass, s_Property, veht_auto)
If m_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(m_rst("Prop_Rate_Auto")) Then
    Call UpdateRstField(m_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
End If
Fat_Rate_Truck = LookupFatalityRate(m_dbs, hclass, s_Fatality, veht_truck)
If m_rst("Fat_Rate_Auto") <> Fat_Rate_Truck Or IsNull(m_rst("Fat_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "Fat_Rate_Truck", Fat_Rate_Truck)
End If
Inj_Rate_Truck = LookupFatalityRate(m_dbs, hclass, s_Injury, veht_truck)
If m_rst("Inj_Rate_Truck") <> Inj_Rate_Truck Or IsNull(m_rst("Inj_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "Inj_Rate_Truck", Inj_Rate_Truck)
End If
Prop_Rate_Truck = LookupFatalityRate(m_dbs, hclass, s_Property, veht_truck)
If m_rst("Prop_Rate_Truck") <> Prop_Rate_Truck Or IsNull(m_rst("Prop_Rate_Truck")) Then
    Call UpdateRstField(m_rst, "Prop_Rate_Truck", Prop_Rate_Truck)
End If
Fatality_Auto = VMTAuto_I * NPVF_VMTAuto * Fat_Rate_Auto / 100000000
Injury_Auto = VMTAuto_I * NPVF_VMTAuto * Inj_Rate_Auto / 1000000
Property_Auto = VMTAuto_I * NPVF_VMTAuto * Prop_Rate_Auto / 1000000
Fatality_Fr = FrVMT_I * NPVF_FrVMT * Fat_Rate_Truck / 100000000
Injury_Fr = FrVMT_I * NPVF_FrVMT * Inj_Rate_Truck / 1000000
Property_Fr = FrVMT_I * NPVF_FrVMT * Prop_Rate_Truck / 1000000
Fatality_Bus = VMTBus_I * NPVF_VMTBus * m_rst("Fat_Rate_Bus") / 100000000
Injury_Bus = VMTBus_I * NPVF_VMTBus * m_rst("Inj_Rate_Bus") / 1000000
Property_Bus = VMTBus_I * NPVF_VMTBus * m_rst("Prop_Rate_Bus") / 1000000
Fatality_Rail = VMTRail_I * NPVF_VMTRail * m_rst("Fat_Rate_Rail") / 100000000
Injury_Rail = VMTRail_I * NPVF_VMTRail * m_rst("Inj_Rate_Rail") / 1000000
Property_Rail = VMTRail_I * NPVF_VMTRail * m_rst("Prop_Rate_Rail") / 1000000

```

```

m_Fatality = Fatality_Auto + Fatality_Fr + Fatality_Rail + Fatality_Bus
m_Injury = Injury_Auto + Injury_Fr + Injury_Rail + Injury_Bus
m_Property = Property_Auto + Property_Fr + Property_Rail + Property_Bus
'Set module level values (to be displayed)
  m_Values.Add m_Fatality, "FATALITY"
  m_Values.Add m_Injury, "INJURY"
  m_Values.Add m_Property, "PROPERTY"

'Safety Benefit Calculations
NPVF_AutoSafety = NPVF_AutoEnvBen
Fatality_AutoBen = VMTAuto_I * Fat_Rate_Auto / 100000000 * Cost_Fatality * NPVF_AutoSafety
Injury_AutoBen = VMTAuto_I * Inj_Rate_Auto / 1000000 * Cost_Evident * NPVF_AutoSafety
Prop_AutoBen = VMTAuto_I * Prop_Rate_Auto / 1000000 * Cost_PDO * NPVF_AutoSafety
NPVF_FrSafety = NPVF_FrEnvBen
Fatality_FrBen = VMTAuto_I * Fat_Rate_Truck / 100000000 * Cost_Fatality * NPVF_FrSafety
Injury_FrBen = VMTAuto_I * Inj_Rate_Truck / 1000000 * Cost_Evident * NPVF_FrSafety
Prop_FrBen = VMTAuto_I * Prop_Rate_Truck / 1000000 * Cost_PDO * NPVF_FrSafety
NPVF_BusSafety = NPVF_BusEnvBen
Fatality_BusBen = VMTBus_I * NPVF_VMTBus * m_rst("Fat_Rate_Bus") / 100000000 * Cost_Fatality *
NPVF_BusSafety
Injury_BusBen = VMTBus_I * NPVF_VMTBus * m_rst("Inj_Rate_Bus") / 1000000 * Cost_Evident *
NPVF_BusSafety
Property_BusBen = VMTBus_I * NPVF_VMTBus * m_rst("Prop_Rate_Bus") / 1000000 * Cost_PDO *
NPVF_BusSafety
NPVF_RailSafety = NPVF_RailEnvBen
Fatality_RailBen = VMTRail_I * NPVF_VMTRail * m_rst("Fat_Rate_Rail") / 100000000 * Cost_Fatality *
NPVF_RailSafety
Injury_RailBen = VMTRail_I * NPVF_VMTRail * m_rst("Inj_Rate_Rail") / 1000000 * Cost_Evident *
NPVF_RailSafety
Property_RailBen = VMTRail_I * NPVF_VMTRail * m_rst("Prop_Rate_Rail") / 1000000 * Cost_PDO *
NPVF_RailSafety

Safety_Ben = Fatality_AutoBen + Injury_AutoBen + Prop_AutoBen + Fatality_FrBen + Injury_FrBen + _
Prop_FrBen + Fatality_RailBen + Injury_RailBen + Prop_RailBen + _
Fatality_BusBen + Injury_BusBen + Prop_BusBen
'Set module level values (to be displayed)
  m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
tCost = Total_Cost + OpMaint_Cost - Terminal_Cost - EnvRetrofit_Cost
If tCost <> 0 Then
  m_BCR = (m_TT_Ben + m_User_Ben + m_Safety_Ben + m_Env_Ben) / tCost
End If
m_WSDOT_BCR = (m_TT_Ben + m_User_Ben + m_Safety_Ben + m_Env_Ben) / (WSDOT_Cost +
OpMaint - EnvRetrofit)

'Set module level values (to be displayed)
  m_Values.Add m_Total_Cost, "TOTAL_COST"
  m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
  m_Values.Add m_Federal_Cost, "FEDERAL_COST"
  m_Values.Add m_BCR, "BCR"
  m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

End Sub

Private Sub CalculateOutObjScores()

'Calculation for the System Operation and Maintenance
  m_Sys_OM = (m_rst("Q1A") * 34) + (33 * m_rst("Q1B")) + (33 * m_rst("Q1D"))
  m_Values.Add CInt(m_Sys_OM), "SYS_OM"
'Calculation for the System Preservation
  m_Sys_Pres = 100 * m_rst("Q2A")

```



```

    m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
    m_Sp_Needs = 100 * m_rst("Q3A")
    m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
    If m_rst("WTP_Corridor") Then
        m_Cong_Rel = m_Values.item("TT_BEN")
    Else
        m_Cong_Rel = 0
    End If
    m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
    m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)
    m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
    m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
    m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
    ' This must be updated later
    m_Safety = m_Safety_Ben
    m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
    m_Commnty = ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B"))) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E"))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") > 0 Then
        m_Collab = 50
    Else
        m_Collab = 0
    End If
    m_Collab = m_Collab + (50 * m_rst("Q10A"))
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    ' This must be updated later
    m_Air_Qual = m_Env_Ben
    m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = Clnt(((25 * m_rst("Q16A")) + (25 * m_rst("Q16B"))) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

```

End Sub

Public Function GetItemValue(itemname As String) As Variant

Dim rtnval

rtnval = m_Values(itemname)

GetItemValue = rtnval

End Function

Public Function DisplayFormat(fld As String, dval As Variant) As Variant

Dim crncylist As String

Dim intlist As String

Dim dbllist As String

'String to list the fields with currency formats

crncylist = "TT_BEN,USER_BEN,USER_TRANSFER,ENV_BEN,SAFETY_BEN," & _
"TOTAL_COST,WSDOT_COST,FEDERAL_COST"

intlist = "TT_MIN,SYS_OM,SYS_PRES,SP_NEEDS,CONG_REL,TRAV_OPT,SEAMLESS," & _
"O_SAFETY,SECURITY,COMMNTY,COLLAB,FREIGHT,ECON_PROS," & _
"TOURISM,AIR_QUAL,WTR_QUAL,HABITAT,RESOURCE"

dbllist = "FATALITY,INJURY,PROPERTY,CO_TONS,VOC_TONS,NOX_TONS," & _
"PM10_TONS,BCR,WSDOT_BCR"

If InStr(crncylist, fld) <> 0 Then

If dval <> 0 Then

dval = Format(dval, "\$#,###")

Else

dval = "\$0"

End If

Elseif InStr(intlist, fld) <> 0 Then

dval = Round(dval, 0)

Elseif InStr(dbllist, fld) <> 0 Then

dval = Round(dval, 2)

End If

DisplayFormat = dval

End Function

Private Function UpdateRstField(ByRef m_rst As DAO.Recordset, fld As String, fldval As Variant) As Boolean

On Error Resume Next

'Initialize return value

UpdateRstField = True

'Edit the fields value

m_rst.Edit

m_rst(fld) = fldval

m_rst.Update

'If there was an error clear it and return false to indicate that nothing occurred

If Err <> 0 Then

UpdateRstField = False

Err.Clear

End If

End Function

Private Sub Class_Terminate()

Set m_dbsMICA = Nothing

Set m_rstfryOO = Nothing

Set m_Values = Nothing

End Sub

Program Code for Transportation Demand Management

Areawide Programs

Option Compare Database

'Variables for the class level database objects

```
Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection
```

'Variables used in intermediate calculations (variants)

```
Dim Forecast_Period
Dim m_VMT_Shift
Dim TDM_Reduction
Dim m_Approach
Dim m_WSDOT_Cap
Dim m_Federal_Cap
Dim m_Other1_Cap
Dim m_Other2_Cap
Dim m_Other3_Cap
Dim m_WSDOT_OM
Dim m_Federal_OM
Dim m_Other1_OM
Dim m_Other2_OM
Dim m_Other3_OM
```

'Variables to hold the calculated values (variants)

```
Dim m_Total_Benefit
Dim m_OpCost
Dim m_Pricing_Rev
Dim m_TT_Ben
Dim m_User_Ben
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
Dim m_Other2_Cost
Dim m_Other3_Cost
Dim m_Cap_Cost
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
```

```
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
'Speed Assumed for lookup table
Private Const AUTOSPEED = 35
```

```
Private Sub Class_Initialize()
    m_qryName = "tdm_calc_Areawide"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If
```

```

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")

'Calculate economic analysis factors
Discount_Rate = a_rst("Discount_Rate")
PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
(Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - _
(Forecast_Period / ((1 + Discount_Rate) ^ Forecast_Period)))
PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Calculate TDM Reduction Factor
TDM_Reduction = m_rst("vanpool_red") + m_rst("info_red") + _
m_rst("employer_red") + m_rst("landuse_red") + m_rst("service_red") + _
m_rst("other_red") + m_rst("pricing_red")

'Estimate VMT shift due to proposed project
Vmt_shift_i = TDM_Reduction * m_rst("BC_ann_VMT_i")

```

```

Vmt_shift_f = TDM_Reduction * m_rst("BC_ann_VMT_f")

m_VMT_Shift = (Vmt_shift_i + Vmt_shift_f) * Forecast_Period / 2

'Set module level values (to be displayed)
  m_Values.Add Forecast_Period, "FORECAST_PERIOD"
  m_Values.Add TDM_Reduction, "TDM_REDUCTION"
  m_Values.Add m_VMT_Shift, "VMT_SHIFT"

'Travel Time Savings Calculations
'No travel time benefits are calculated for this project type
m_TT_Ben = 0

'Operating Cost Savings Calculations
Veh_OpCost_Full = a_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = a_rst("Veh_OpCost_Direct")
If a_rst("Full_Cost") Then
  m_Approach = "Full cost"
  OpCost_i = Vmt_shift_i * Veh_OpCost_Full
  OpCost_f = Vmt_shift_f * Veh_OpCost_Full
Else
  m_Approach = "Direct cost"
  OpCost_i = Vmt_shift_i * Veh_OpCost_Direct
  OpCost_f = Vmt_shift_f * Veh_OpCost_Direct
End If

G_Opcost = (OpCost_f - OpCost_i) / (Forecast_Period - 1)
m_OpCost = (OpCost_i * PofA) + (G_Opcost * PofG)

'Out-of-pocket costs to users due to pricing
price_rev_i = m_rst("price_rev_i")
price_rev_f = m_rst("price_rev_f")
G_Price_rev = (price_rev_f - price_rev_i) / (Forecast_Period - 1)
m_Pricing_Rev = -((price_rev_i * PofA) + (G_Price_rev * PofG))

'User Benefit NPV Calculation
m_User_Ben = m_OpCost + m_Pricing_Rev

'Set module level values (to be displayed)
  m_Values.Add m_Approach, "APPROACH"
  m_Values.Add m_OpCost, "OPCOST"
  m_Values.Add m_Pricing_Rev, "PRICING_REV"
  m_Values.Add m_User_Ben, "USER_BEN"
  m_Values.Add m_TT_Ben, "TT_BEN"

'Air Pollution Calculations - for diverted auto trips
Per_Cold_Auto = a_rst("Per_Cold_Auto")
CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)

If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
End If
CO_Cold_Auto = a_rst("CO_Cold_Auto")
COTon_Cost = a_rst("COTon_Cost")
VOC_Rate_Auto = a_rst("VOC_Rate_Auto")

```

```

VOC_Cold_Auto = a_rst("VOC_Cold_Auto")
VOCTon_Cost = a_rst("VOCTon_Cost")
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
End If
NOX_Cold_Auto = a_rst("NOX_Cold_Auto")
NOXTon_Cost = a_rst("NOXTon_Cost")
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")
PM10_Cold_Auto = a_rst("PM10_Cold_Auto")
PM10Ton_Cost = a_rst("PM10Ton_Cost")

```

```

CO_Tons_i = (Vmt_shift_i * CO_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
CO_Tons_f = (Vmt_shift_f * CO_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
VOC_Tons_i = (Vmt_shift_i * VOC_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
VOC_Tons_f = (Vmt_shift_f * VOC_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
NOX_Tons_i = (Vmt_shift_i * NOX_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
NOX_Tons_f = (Vmt_shift_f * NOX_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
PM10_Tons_i = (Vmt_shift_i * PM10_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)
PM10_Tons_f = (Vmt_shift_f * PM10_Rate_Auto) * (1 / 1000) * (0.9842 / 1000)

```

'Emission Sums

```

m_CO_Tons = (-1) * (CO_Tons_f + CO_Tons_i) * Forecast_Period / 2
m_VOC_Tons = (-1) * (VOC_Tons_f + VOC_Tons_i) * Forecast_Period / 2
m_NOX_Tons = (-1) * (NOX_Tons_f + NOX_Tons_i) * Forecast_Period / 2
m_PM10_Tons = (-1) * (PM10_Tons_f + PM10_Tons_i) * Forecast_Period / 2

```

'Emission Benefit Calculations

```

EnvBen_i = CO_Tons_i * COTon_Cost + VOC_Tons_i * VOCTon_Cost + NOX_Tons_i *

```

```

- NOXTon_Cost + PM10_Tons_i * PM10Ton_Cost

```

```

EnvBen_f = CO_Tons_f * COTon_Cost + VOC_Tons_f * VOCTon_Cost + NOX_Tons_f *

```

```

- NOXTon_Cost + PM10_Tons_f * PM10Ton_Cost

```

```

G_EnvBen = (EnvBen_f - EnvBen_i) / (Forecast_Period - 1)

```

'Emission NPV Calculations

```

m_Env_Ben = (EnvBen_i * PofA) + (G_EnvBen * PofG)

```

'Set module level values (to be displayed)

```

m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"
m_Values.Add m_PM10_Tons, "PM10_TONS"
m_Values.Add m_Env_Ben, "ENV_BEN"

```

'Safety Calculations

```

Evident_Cost = a_rst("Evident_Cost")
Fatality_Cost = a_rst("Fatality_Cost")
PDO_Cost = a_rst("PDO_Cost")

```

'Look up accident rates from lookup table

```

Fat_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_auto)
If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then

```

```

    Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
End If
Inj_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Injury, veht_auto)
If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
End If
Prop_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Property, veht_auto)
If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
End If

'Accident reduction due to shifted vmt
auto_Fatality_i = Vmt_shift_i * Fat_Rate_Auto / 1000000
auto_Fatality_f = Vmt_shift_f * Fat_Rate_Auto / 1000000
auto_Injury_i = Vmt_shift_i * Inj_Rate_Auto / 1000000
auto_Injury_f = Vmt_shift_f * Inj_Rate_Auto / 1000000
auto_Property_i = Vmt_shift_i * Prop_Rate_Auto / 1000000
auto_Property_f = Vmt_shift_f * Prop_Rate_Auto / 1000000
m_Fatality = (-1) * (auto_Fatality_i + auto_Fatality_f) * (Forecast_Period) / 2
m_Injury = (-1) * (auto_Injury_i + auto_Injury_f) * (Forecast_Period) / 2
m_Property = (-1) * (auto_Property_i + auto_Property_f) * (Forecast_Period) / 2

'Calculate safety benefits
Safety_Ben_i = (auto_Fatality_i * Fatality_Cost) + _
(auto_Injury_i * Evident_Cost) + (auto_Property_i * PDO_Cost)
Safety_Ben_f = (auto_Fatality_f * Fatality_Cost) + _
(auto_Injury_f * Evident_Cost) + (auto_Property_f * PDO_Cost)
G_safety = (Safety_Ben_f - Safety_Ben_i) / (Forecast_Period - 1)
m_Safety_Ben = (Safety_Ben_i * PofA) + (G_safety * PofG)

'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
'Capital Cost
For i = 1 To 5
    m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
    m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
        1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA

```



```

m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM

```

```

'Terminal cost
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF

```

```

'Total Costs
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
m_Other3_Cost = m_Other3_Cap + m_Other3_OM

```

```

m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
m_Other1_Cost + m_Other2_Cost + m_Other3_Cost

```

```

Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)

```

```

'Environmental Retrofit Calculations
fishbarrier_bc = a_rst("fishbarrier_bc")
stormwater_bc = a_rst("stormwater_bc")
noisebarrier_bc = a_rst("noisebarrier_bc")

```

```

'Environmental Retrofit Costs

```

```

For i = 1 To 5

```

```

m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))

```

```

Next

```

```

m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap

```

```

'Environmental Retrofit Benefits

```

```

m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc

```

```

m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben

```

```

'Benefit-Cost Calculations

```

```

Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben

```

```

If (m_Total_Cost - m_Terminal_Cost) = 0 Then

```

```

m_BCR = 0

```

```

Else

```

```

m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)

```

```

End If

```

```

If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then

```

```

m_WSDOT_BCR = 0

```

```

Else

```

```

m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)

```

```

End If
'Set module level values (to be displayed)
  m_Values.Add m_Total_Cost, "TOTAL_COST"
  m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
  m_Values.Add m_Federal_Cost, "FEDERAL_COST"
  m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
  m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
  m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
  m_Values.Add m_Other1_Cost, "OTHER1_COST"
  m_Values.Add m_Other2_Cost, "OTHER2_COST"
  m_Values.Add m_Other3_Cost, "OTHER3_COST"
  m_Values.Add m_Cap_Cost, "CAP_COST"
  m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
  m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
  m_Values.Add m_Other1_Cap, "OTHER1_CAP"
  m_Values.Add m_Other2_Cap, "OTHER2_CAP"
  m_Values.Add m_Other3_Cap, "OTHER3_CAP"
  m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
  m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
  m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
  m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
  m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
  m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
  m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
  m_Values.Add m_WSDOT_OM, "WSDOT_OM"
  m_Values.Add m_Federal_OM, "FEDERAL_OM"
  m_Values.Add m_Other1_OM, "OTHER1_OM"
  m_Values.Add m_Other2_OM, "OTHER2_OM"
  m_Values.Add m_Other3_OM, "OTHER3_OM"
  m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
  m_Values.Add m_BCR, "BCR"
  m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

On Error Resume Next

```

'Calculation for the System Operation and Maintenance
  m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)
  m_Values.Add m_Sys_OM, "SYS_OM"
'Calculation for the System Preservation
  m_Sys_Pres = 100 * m_rst("Q2A")
  m_Values.Add m_Sys_Pres, "SYS_PRES"
'Calculation for the Special Needs Transportation
  m_Sp_Needs = 100 * m_rst("Q3A")
  m_Values.Add m_Sp_Needs, "SP_NEEDS"
'Calculation for the Congestion Relief
  If m_rst("WTP_Corridor") Then
    m_Cong_Rel = 50 + (m_rst("Q4") * 50)
  Else
    m_Cong_Rel = (m_rst("Q4") * 50)
  End If
  m_Values.Add m_Cong_Rel, "CONG_REL"
'Calculation for Increased Travel Options
  m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)

```

```

    m_Values.Add m_Trav_Opt, "TRAV_OPT"
'Calculation for Seamless Connections
    m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)
    m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
    If m_Safety_Ben > 0 Then
        m_Safety = 50 + (m_rst("Q7B") * 50)
    Else
        m_Safety = m_rst("Q7B") * 50
    End If
    m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
    m_Security = 100 * m_rst("Q8A")
    m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
    'The -1 Multiplication Corrects the negative sign introduced for true
    m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
        (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
        (20 * m_rst("Q9E")))
    m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
    If m_rst("Q10B") < 5 Then
        m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
    Else
        m_Collab = (m_rst("Q10A") * 50) + 50
    End If
    m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
    m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
    m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
    m_Econ_Pro = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
    m_Values.Add m_Econ_Pro, "ECON_PROS"
'Calculation for Tourism
    m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
    m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
    If m_rst("Q14B") = 1 Then
        m_Air_Qual = 50 + m_rst("Q14A") * 50
    Else
        m_Air_Qual = 0
    End If
    m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
    m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
        (33 * m_rst("Q15C"))
    m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
    m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
        (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
        (1 + m_rst("Q16E") + m_rst("Q16F"))
    m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
    m_Resource = 100 * m_rst("Q17")
    m_Values.Add m_Resource, "RESOURCE"

```

End Sub

Public Function GetItemValue(itemname As String) As Variant

 Dim rtnval

 rtnval = m_Values.Retrieve(itemname)

 GetItemValue = rtnval

End Function

Private Sub Class_Terminate()

 a_rst.Close

 m_rst.Close

 m_dbs.Close

 Set a_rst = Nothing

 Set m_rst = Nothing

 Set m_dbs = Nothing

 Set m_Values = Nothing

End Sub

Commuter Trip Reduction Support

Option Compare Database

'Variables for the class level database objects

```
Dim m_dbs As DAO.Database
Dim m_rst As DAO.Recordset
Dim a_rst As DAO.Recordset
Dim m_qryName As String
Dim m_calcsComplete As Boolean
Dim m_Values As ValueCollection
```

'Variables used in intermediate calculations (variants)

```
Dim Forecast_Period
Dim m_Auto_Vmt_Shift
Dim m_Auto_Trip_Shift
Dim m_Transit_Trip_Shift
Dim m_OpCost
Dim m_Fare_Cost
Dim m_Approach
Dim m_WSDOT_Cap
Dim m_Federal_Cap
Dim m_Other1_Cap
Dim m_Other2_Cap
Dim m_Other3_Cap
Dim m_WSDOT_OM
Dim m_Federal_OM
Dim m_Other1_OM
Dim m_Other2_OM
Dim m_Other3_OM
```

'Variables to hold the calculated values (variants)

```
Dim m_Total_Benefit
Dim m_TT_Ben
Dim m_User_Ben
Dim m_CO_Tons
Dim m_VOC_Tons
Dim m_NOX_Tons
Dim m_PM10_Tons
Dim m_Env_Ben
Dim m_Fatality
Dim m_Injury
Dim m_Property
Dim m_Safety_Ben
Dim m_Total_Cost
Dim m_WSDOT_Cost
Dim m_Federal_Cost
Dim m_Other1_Cost
Dim m_Other2_Cost
Dim m_Other3_Cost
Dim m_Cap_Cost
Dim m_OpMaint_Cost
Dim m_Terminal_Cost
```

```
Dim m_BCR
Dim m_WSDOT_BCR
```

```
'Variables to hold the outcome objective scores (variants)
```

```
Dim m_Sys_OM
Dim m_Sys_Pres
Dim m_Sp_Needs
Dim m_Cong_Rel
Dim m_Trav_Opt
Dim m_Seamless
Dim m_Safety
Dim m_Security
Dim m_Commnty
Dim m_Collab
Dim m_Freight
Dim m_Econ_Proc
Dim m_Tourism
Dim m_Air_Qual
Dim m_Wtr_Qual
Dim m_Habitat
Dim m_Resource
```

```
'Speed Assumed for lookup table
Private Const AUTOSPEED = 35
```

```
Private Sub Class_Initialize()
    m_qryName = "tdm_calc_CTR_Support"
    m_calcsComplete = False
End Sub
```

```
Public Function CalculateProjectType(pid As Integer, Optional asmptnID As Integer) As Boolean
```

```
    Dim atype As String
    Dim qryDef As DAO.QueryDef
```

```
    Set m_dbs = CurrentDb
```

```
    ' Open QueryDef object with one parameters.
    Set qryDef = m_dbs.QueryDefs(m_qryName)
    qryDef.Parameters!projID = pid
```

```
    ' Set recordset to new values.
    Set m_rst = qryDef.OpenRecordset
    ' If data is entered for this particular project then calculate all scores
    If Not m_rst.EOF Then
```

```
        'Initialize new values collection to store calculated values
        Set m_Values = New ValueCollection
```

```
        'If you are running MICA use the scenario assumptions
        If IsNull(asmptnID) Or IsEmpty(asmptnID) Or asmptnID = 0 Then
            atype = "prj_Project_Assumptions"
        Else
            atype = "prj_Global_Assumptions"
            pid = asmptnID
        End If
```

```

'Open up a assumption recordset - depend on if in MICA or not
Set qryDef = m_dbs.QueryDefs(atype)
qryDef.Parameters!asmptnID = pid

' Set recordset to new values.
Set a_rst = qryDef.OpenRecordset

'Calculate all of the values for the project type
CalculateBenefits
CalculateOutObjScores

'Check to see if the calcs are done for this project and set input status accordingly
If Not IsArray(g_VerificationFlds) Then SetVerificationFlds
m_calcsComplete = m_Values.VerifyFieldValues(g_VerificationFlds)

Call UpdateRstField(m_rst, "Input_Status", m_calcsComplete)

End If

'Close the Querydef object
qryDef.Close
Set qryDef = Nothing

CalculateProjectType = m_calcsComplete

End Function

Public Property Get InputStatus() As Boolean
Set InputStatus = m_calcsComplete
End Property

Public Property Get ReturnValueCollection() As ValueCollection
Set ReturnValueCollection = m_Values
End Property

Private Sub CalculateBenefits()
On Error Resume Next
'Calculate Forecast Period
Forecast_Period = m_rst("Fore_Year") - m_rst("Init_Year")

'Calculate economic analysis factors
Discount_Rate = a_rst("Discount_Rate")
PofA = (1 - (1 + Discount_Rate) ^ (-Forecast_Period)) / Discount_Rate
PofG = (1 / Discount_Rate) * (((1 + Discount_Rate) ^ Forecast_Period - 1) / _
(Discount_Rate * (1 + Discount_Rate) ^ Forecast_Period)) - (Forecast_Period / _
((1 + Discount_Rate) ^ Forecast_Period))
PofF = (1 + Discount_Rate) ^ (-Forecast_Period)

'Calculate CTR Effectiveness
Ann_Daily_Benefit = a_rst("Ann_Daily_Benefit")
Base_VMT = m_rst("Base_VMT")
Base_Veh_Trips = m_rst("Base_Veh_Trips")
eval_period = m_rst("Eval_period_f") - m_rst("Eval_period_i")
auto_vmt_eff = (m_rst("d_vmt_drive") + m_rst("d_vmt_carpl") + m_rst("d_vmt_vanpl")) /
Base_VMT

```

```

transit_vmt_eff = m_rst("d_vmt_transit") / Base_VMT
auto_trip_eff = (m_rst("d_veh_drive") + m_rst("d_veh_carpl") + m_rst("d_veh_vanpl")) /
(Base_Veh_Trips)
transit_trip_eff = m_rst("d_veh_transit") / (Base_Veh_Trips)

```

```

'Estimate commute trip mode shifts due to proposed project
ann_d_auto_vmt_i = auto_vmt_eff * m_rst("BC_wkday_VMT_i") * Ann_Daily_Benefit
ann_d_transit_vmt_i = transit_vmt_eff * m_rst("BC_wkday_VMT_i") * Ann_Daily_Benefit
Ann_d_auto_trip_i = auto_trip_eff * m_rst("BC_wkday_trips_i") * Ann_Daily_Benefit
ann_d_transit_trip_i = transit_trip_eff * m_rst("BC_wkday_trips_i") * Ann_Daily_Benefit

```

```

ann_d_auto_vmt_f = auto_vmt_eff * m_rst("BC_wkday_VMT_f") * Ann_Daily_Benefit
ann_d_transit_vmt_f = transit_vmt_eff * m_rst("BC_wkday_VMT_f") * Ann_Daily_Benefit
Ann_d_auto_trip_f = auto_trip_eff * m_rst("BC_wkday_trips_f") * Ann_Daily_Benefit
ann_d_transit_trip_f = transit_trip_eff * m_rst("BC_wkday_trips_f") * Ann_Daily_Benefit

```

```

m_Auto_Vmt_Shift = (ann_d_auto_vmt_i + ann_d_auto_vmt_f) * Forecast_Period / 2
m_Auto_Trip_Shift = (Ann_d_auto_trip_i + Ann_d_auto_trip_f) * Forecast_Period / 2
m_Transit_Vmt_Shift = (ann_d_transit_vmt_i + ann_d_transit_vmt_f) * Forecast_Period / 2
m_Transit_Trip_Shift = (ann_d_transit_trip_i + ann_d_transit_trip_f) * Forecast_Period / 2

```

```

'Set module level values (to be displayed)
m_Values.Add m_Auto_Vmt_Shift, "AUTO_VMT_SHIFT"
m_Values.Add m_Auto_Trip_Shift, "AUTO_TRIP_SHIFT"
m_Values.Add m_Transit_Vmt_Shift, "TRANSIT_VMT_SHIFT"
m_Values.Add m_Transit_Trip_Shift, "TRANSIT_TRIP_SHIFT"

```

```

'Travel Time Savings Calculations
'No travel time benefits are calculated for this project type
m_TT_Ben = 0

```

```

'Operating Cost Savings Calculations
Veh_OpCost_Full = a_rst("Veh_OpCost_Full")
Veh_OpCost_Direct = a_rst("Veh_OpCost_Direct")
If a_rst("Full_Cost") Then
    m_Approach = "Full cost"
    OpCost_i = (-1) * ann_d_auto_vmt_i * Veh_OpCost_Full
    OpCost_f = (-1) * ann_d_auto_vmt_f * Veh_OpCost_Full
Else
    m_Approach = "Direct cost"
    OpCost_i = (-1) * ann_d_auto_vmt_i * Veh_OpCost_Direct
    OpCost_f = (-1) * ann_d_auto_vmt_f * Veh_OpCost_Direct
End If

```

```

G_Opcost = (OpCost_f - OpCost_i) / (Forecast_Period - 1)
m_OpCost = (OpCost_i * PofA) + (G_Opcost * PofG)

```

```

'Change in out-of-pocket fare cost of transit trips
Transit_fare = m_rst("Transit_fare")
fare_cost_i = ann_d_transit_trip_i * Transit_fare
fare_cost_f = ann_d_transit_trip_f * Transit_fare
G_fare_cost = (fare_cost_f - fare_cost_i) / (Forecast_Period - 1)
m_Fare_Cost = (-1) * ((fare_cost_i * PofA) + (G_fare_cost * PofG))

```

```

'User Benefit NPV Calculation
m_User_Ben = m_OpCost + m_Fare_Cost

```



```
'Set module level values (to be displayed)
  m_Values.Add m_Approach, "APPROACH"
  m_Values.Add m_OpCost, "OPCOST"
  m_Values.Add m_Fare_Cost, "FARE_COST"
  m_Values.Add m_User_Ben, "USER_BEN"
  m_Values.Add m_TT_Ben, "TT_BEN"
```

```
'Air Pollution Calculations - for diverted auto trips
```

```
Per_Cold_Auto = a_rst("Per_Cold_Auto")
CO_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_CO, veht_auto)
```

```
If a_rst("CO_Rate_Auto") <> CO_Rate_Auto Or IsNull(a_rst("CO_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "CO_Rate_Auto", CO_Rate_Auto)
```

```
End If
```

```
CO_Cold_Auto = a_rst("CO_Cold_Auto")
```

```
COTon_Cost = a_rst("COTon_Cost")
```

```
VOC_Rate_Auto = a_rst("VOC_Rate_Auto")
```

```
VOC_Cold_Auto = a_rst("VOC_Cold_Auto")
```

```
VOCTon_Cost = a_rst("VOCTon_Cost")
```

```
NOX_Rate_Auto = LookupAirPollution(m_dbs, AUTOSPEED, em_NOX, veht_auto)
```

```
If a_rst("NOX_Rate_Auto") <> NOX_Rate_Auto Or IsNull(a_rst("NOX_Rate_Auto")) Then
  Call UpdateRstField(a_rst, "NOX_Rate_Auto", NOX_Rate_Auto)
```

```
End If
```

```
NOX_Cold_Auto = a_rst("NOX_Cold_Auto")
```

```
NOXTon_Cost = a_rst("NOXTon_Cost")
```

```
PM10_Rate_Auto = a_rst("PM10_Rate_Auto")
```

```
PM10_Cold_Auto = a_rst("PM10_Cold_Auto")
```

```
PM10Ton_Cost = a_rst("PM10Ton_Cost")
```

```
CO_Tons_i = ((ann_d_auto_vmt_i * CO_Rate_Auto) + _
  (Ann_d_auto_trip_i * Per_Cold_Auto * CO_Cold_Auto)) * _
  (1 / 1000) * (0.9842 / 1000)
```

```
CO_Tons_f = ((ann_d_auto_vmt_f * CO_Rate_Auto) + _
  (Ann_d_auto_trip_f * Per_Cold_Auto * CO_Cold_Auto)) * _
  (1 / 1000) * (0.9842 / 1000)
```

```
VOC_Tons_i = ((ann_d_auto_vmt_i * VOC_Rate_Auto) + _
  (Ann_d_auto_trip_i * Per_Cold_Auto * VOC_Cold_Auto)) * _
  (1 / 1000) * (0.9842 / 1000)
```

```
VOC_Tons_f = ((ann_d_auto_vmt_f * VOC_Rate_Auto) + _
  (Ann_d_auto_trip_f * Per_Cold_Auto * VOC_Cold_Auto)) * _
  (1 / 1000) * (0.9842 / 1000)
```

```
NOX_Tons_i = ((ann_d_auto_vmt_i * NOX_Rate_Auto) + _
  (Ann_d_auto_trip_i * Per_Cold_Auto * NOX_Cold_Auto)) * _
  (1 / 1000) * (0.9842 / 1000)
```

```
NOX_Tons_f = ((ann_d_auto_vmt_f * NOX_Rate_Auto) + _
  (Ann_d_auto_trip_f * Per_Cold_Auto * NOX_Cold_Auto)) * _
  (1 / 1000) * (0.9842 / 1000)
```

```
PM10_Tons_i = ((ann_d_auto_vmt_i * PM10_Rate_Auto) + _
  (Ann_d_auto_trip_i * Per_Cold_Auto * PM10_Cold_Auto)) * _
  (1 / 1000) * (0.9842 / 1000)
```

```
PM10_Tons_f = ((ann_d_auto_vmt_f * PM10_Rate_Auto) + _
  (Ann_d_auto_trip_f * Per_Cold_Auto * PM10_Cold_Auto)) * _
  (1 / 1000) * (0.9842 / 1000)
```

```
'Emission Sums
```

```

m_CO_Tons = (CO_Tons_f + CO_Tons_i) * Forecast_Period / 2
m_VOC_Tons = (VOC_Tons_f + VOC_Tons_i) * Forecast_Period / 2
m_NOX_Tons = (NOX_Tons_f + NOX_Tons_i) * Forecast_Period / 2
m_PM10_Tons = (PM10_Tons_f + PM10_Tons_i) * Forecast_Period / 2

```

'Emission Benefit Calculations

```

EnvBen_i = (-1) * (CO_Tons_i * COTon_Cost + VOC_Tons_i * VOCTon_Cost +
NOX_Tons_i * NOXTon_Cost + PM10_Tons_i * PM10Ton_Cost)
EnvBen_f = (-1) * (CO_Tons_f * COTon_Cost + VOC_Tons_f * VOCTon_Cost +
NOX_Tons_f * NOXTon_Cost + PM10_Tons_f * PM10Ton_Cost)
G_EnvBen = (EnvBen_f - EnvBen_i) / (Forecast_Period - 1)

```

'Emission NPV Calculations

```

m_Env_Ben = (EnvBen_i * PofA) + (G_EnvBen * PofG)

```

'Set module level values (to be displayed)

```

m_Values.Add m_CO_Tons, "CO_TONS"
m_Values.Add m_VOC_Tons, "VOC_TONS"
m_Values.Add m_NOX_Tons, "NOX_TONS"
m_Values.Add m_PM10_Tons, "PM10_TONS"
m_Values.Add m_Env_Ben, "ENV_BEN"

```

'Safety Calculations

```

Fat_Rate_Bus = a_rst("Fat_Rate_Bus")
Inj_Rate_Bus = a_rst("Inj_Rate_Bus")
Prop_Rate_Bus = a_rst("Prop_Rate_Bus")
Evident_Cost = a_rst("Evident_Cost")
Fatality_Cost = a_rst("Fatality_Cost")
PDO_Cost = a_rst("PDO_Cost")

```

'Look up accident rates from lookup table

```

Fat_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Fatality, veht_auto)
If a_rst("Fat_Rate_Auto") <> Fat_Rate_Auto Or IsNull(a_rst("Fat_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Fat_Rate_Auto", Fat_Rate_Auto)
End If
Inj_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Injury, veht_auto)
If a_rst("Inj_Rate_Auto") <> Inj_Rate_Auto Or IsNull(a_rst("Inj_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Inj_Rate_Auto", Inj_Rate_Auto)
End If
Prop_Rate_Auto = LookupFatalityRate(m_dbs, 2, s_Property, veht_auto)
If a_rst("Prop_Rate_Auto") <> Prop_Rate_Auto Or IsNull(a_rst("Prop_Rate_Auto")) Then
    Call UpdateRstField(a_rst, "Prop_Rate_Auto", Prop_Rate_Auto)
End If

```

'Accident reduction due to shifted auto vmt

```

auto_Fatality_i = ann_d_auto_vmt_i * Fat_Rate_Auto / 1000000
auto_Fatality_f = ann_d_auto_vmt_f * Fat_Rate_Auto / 1000000
auto_Injury_i = ann_d_auto_vmt_i * Inj_Rate_Auto / 1000000
auto_Injury_f = ann_d_auto_vmt_f * Inj_Rate_Auto / 1000000
auto_Property_i = ann_d_auto_vmt_i * Prop_Rate_Auto / 1000000
auto_Property_f = ann_d_auto_vmt_f * Prop_Rate_Auto / 1000000
auto_Fatality = (auto_Fatality_i + auto_Fatality_f) * (Forecast_Period) / 2
auto_Injury = (auto_Injury_i + auto_Injury_f) * (Forecast_Period) / 2
auto_Property = (auto_Property_i + auto_Property_f) * (Forecast_Period) / 2

```

```

'Accident reduction due to shifted transit vmt
transit_Fatality_i = ann_d_transit_vmt_i * Fat_Rate_Bus / 1000000
transit_Fatality_f = ann_d_transit_vmt_f * Fat_Rate_Bus / 1000000
transit_Injury_i = ann_d_transit_vmt_i * Inj_Rate_Bus / 1000000
transit_Injury_f = ann_d_transit_vmt_f * Inj_Rate_Bus / 1000000
transit_Property_i = ann_d_transit_vmt_i * Prop_Rate_Bus / 1000000
transit_Property_f = ann_d_transit_vmt_f * Prop_Rate_Bus / 1000000
transit_Fatality = (transit_Fatality_i + transit_Fatality_f) * (Forecast_Period) / 2
transit_Injury = (transit_Injury_i + transit_Injury_f) * (Forecast_Period) / 2
transit_Property = (transit_Property_i + transit_Property_f) * (Forecast_Period) / 2

'Total accident reduction
m_Fatality = auto_Fatality + transit_Fatality
m_Injury = auto_Injury + transit_Injury
m_Property = auto_Property + transit_Property

'Calculate safety benefits
Safety_Ben_i = (-1) * (((auto_Fatality_i + transit_Fatality_i) * Fatality_Cost) + _
((auto_Injury_i + transit_Injury_i) * Evident_Cost) + _
((auto_Property_i + transit_Property_i) * PDO_Cost))

Safety_Ben_f = (-1) * (((auto_Fatality_f + transit_Fatality_f) * Fatality_Cost) + _
((auto_Injury_f + transit_Injury_f) * Evident_Cost) + _
((auto_Property_f + transit_Property_f) * PDO_Cost))
G_safety = (Safety_Ben_f - Safety_Ben_i) / (Forecast_Period - 1)
m_Safety_Ben = (Safety_Ben_i * PofA) + (G_safety * PofG)

'Set module level values (to be displayed)
m_Values.Add m_Fatality, "FATALITY"
m_Values.Add m_Injury, "INJURY"
m_Values.Add m_Property, "PROPERTY"
m_Values.Add m_Safety_Ben, "SAFETY_BEN"

'Cost Calculations
'Capital Cost
For i = 1 To 5
m_WSDOT_Cap = m_WSDOT_Cap + (m_rst("WsdotCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Federal_Cap = m_Federal_Cap + (m_rst("FederalCap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other1_Cap = m_Other1_Cap + (m_rst("Other1Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other2_Cap = m_Other2_Cap + (m_rst("Other2Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
m_Other3_Cap = m_Other3_Cap + (m_rst("Other3Cap_Bi" & i) * _
1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
Next
m_Cap_Cost = m_WSDOT_Cap + m_Federal_Cap + m_Other1_Cap + m_Other2_Cap +
m_Other3_Cap

'Operations and Maintenance Cost
m_WSDOT_OM = m_rst("WSDOT_annOM") * PofA
m_Federal_OM = m_rst("Federal_annOM") * PofA
m_Other1_OM = m_rst("Other1_annOM") * PofA
m_Other2_OM = m_rst("Other2_annOM") * PofA
m_Other3_OM = m_rst("Other3_annOM") * PofA

```

```
m_OpMaint_Cost = m_WSDOT_OM + m_Federal_OM + m_Other1_OM + m_Other2_OM +
m_Other3_OM
```

```
'Terminal cost
```

```
m_Terminal_Cost = m_rst("Term_Value_PCF") * PofF
```

```
'Total Costs
```

```
m_WSDOT_Cost = m_WSDOT_Cap + m_WSDOT_OM
```

```
m_Federal_Cost = m_Federal_Cap + m_Federal_OM
```

```
m_Other1_Cost = m_Other1_Cap + m_Other1_OM
```

```
m_Other2_Cost = m_Other2_Cap + m_Other2_OM
```

```
m_Other3_Cost = m_Other3_Cap + m_Other3_OM
```

```
m_Total_Cost = m_WSDOT_Cost + m_Federal_Cost + _
```

```
  m_Other1_Cost + m_Other2_Cost + m_Other3_Cost
```

```
  Call UpdateRstField(m_rst, "Wsdot_TotalCost", m_WSDOT_Cost)
```

```
'Environmental Retrofit Calculations
```

```
  fishbarrier_bc = a_rst("fishbarrier_bc")
```

```
  stormwater_bc = a_rst("stormwater_bc")
```

```
  noisebarrier_bc = a_rst("noisebarrier_bc")
```

```
'Environmental Retrofit Costs
```

```
For i = 1 To 5
```

```
  m_FishBarrier_Cap = m_FishBarrier_Cap + (m_rst("FishBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
```

```
  m_StormWater_Cap = m_StormWater_Cap + (m_rst("StormWater_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
```

```
  m_NoiseBarrier_Cap = m_NoiseBarrier_Cap + (m_rst("NoiseBarrier_Bi" & i) * _
    1 / ((1 + Discount_Rate) ^ (2 * i - 1)))
```

```
Next
```

```
m_EnvRetrofit_Cost = m_FishBarrier_Cap + m_StormWater_Cap + m_NoiseBarrier_Cap
```

```
'Environmental Retrofit Benefits
```

```
  m_FishBarrier_Ben = m_FishBarrier_Cap * fishbarrier_bc
```

```
  m_StormWater_Ben = m_StormWater_Cap * stormwater_bc
```

```
  m_NoiseBarrier_Ben = m_NoiseBarrier_Cap * noisebarrier_bc
```

```
m_EnvRetrofit_Ben = m_FishBarrier_Ben + m_StormWater_Ben + m_NoiseBarrier_Ben
```

```
'Benefit-Cost Calculations
```

```
  Total_Benefit = m_TT_Ben + m_User_Ben + m_Env_Ben + m_Safety_Ben +
m_EnvRetrofit_Ben
```

```
  If (m_Total_Cost - m_Terminal_Cost) = 0 Then
```

```
    m_BCR = 0
```

```
  Else
```

```
    m_BCR = Total_Benefit / (m_Total_Cost - m_Terminal_Cost)
```

```
  End If
```

```
  If (m_WSDOT_Cost - m_Terminal_Cost) = 0 Then
```

```
    m_WSDOT_BCR = 0
```

```
  Else
```

```
    m_WSDOT_BCR = Total_Benefit / (m_WSDOT_Cost - m_Terminal_Cost)
```

```
  End If
```

```
'Set module level values (to be displayed)
```

```
  m_Values.Add m_Total_Cost, "TOTAL_COST"
```

```

m_Values.Add m_WSDOT_Cost, "WSDOT_COST"
m_Values.Add m_Federal_Cost, "FEDERAL_COST"
m_Values.Add m_rst("Other1_Name"), "OTHER1_NAME"
m_Values.Add m_rst("Other2_Name"), "OTHER2_NAME"
m_Values.Add m_rst("Other3_Name"), "OTHER3_NAME"
m_Values.Add m_Other1_Cost, "OTHER1_COST"
m_Values.Add m_Other2_Cost, "OTHER2_COST"
m_Values.Add m_Other3_Cost, "OTHER3_COST"
m_Values.Add m_Cap_Cost, "CAP_COST"
m_Values.Add m_WSDOT_Cap, "WSDOT_CAP"
m_Values.Add m_Federal_Cap, "FEDERAL_CAP"
m_Values.Add m_Other1_Cap, "OTHER1_CAP"
m_Values.Add m_Other2_Cap, "OTHER2_CAP"
m_Values.Add m_Other3_Cap, "OTHER3_CAP"
m_Values.Add m_FishBarrier_Cap, "FISHBARRIER_CAP"
m_Values.Add m_StormWater_Cap, "STORMWATER_CAP"
m_Values.Add m_NoiseBarrier_Cap, "NOISEBARRIER_CAP"
m_Values.Add m_FishBarrier_Ben, "FISHBARRIER_BEN"
m_Values.Add m_StormWater_Ben, "STORMWATER_BEN"
m_Values.Add m_NoiseBarrier_Ben, "NOISEBARRIER_BEN"
m_Values.Add m_OpMaint_Cost, "OPMAINT_COST"
m_Values.Add m_WSDOT_OM, "WSDOT_OM"
m_Values.Add m_Federal_OM, "FEDERAL_OM"
m_Values.Add m_Other1_OM, "OTHER1_OM"
m_Values.Add m_Other2_OM, "OTHER2_OM"
m_Values.Add m_Other3_OM, "OTHER3_OM"
m_Values.Add m_Terminal_Cost, "TERMINAL_COST"
m_Values.Add m_BCR, "BCR"
m_Values.Add m_WSDOT_BCR, "WSDOT_BCR"

```

End Sub

Private Sub CalculateOutObjScores()

'Calculation for the System Operation and Maintenance

$m_Sys_OM = (m_rst("Q1A") * 34) + (m_rst("Q1B") * 33) + (m_rst("Q1D") * 33)$

m_Values.Add m_Sys_OM, "SYS_OM"

'Calculation for the System Preservation

$m_Sys_Pres = 100 * m_rst("Q2A")$

m_Values.Add m_Sys_Pres, "SYS_PRES"

'Calculation for the Special Needs Transportation

$m_Sp_Needs = 100 * m_rst("Q3A")$

m_Values.Add m_Sp_Needs, "SP_NEEDS"

'Calculation for the Congestion Relief

If m_rst("WTP_Corridor") Then

$m_Cong_Rel = 50 + (m_rst("Q4") * 50)$

Else

$m_Cong_Rel = (m_rst("Q4") * 50)$

End If

m_Values.Add m_Cong_Rel, "CONG_REL"

'Calculation for Increased Travel Options

$m_Trav_Opt = (m_rst("Q5A") * 50) + (m_rst("Q5B") * 50)$

m_Values.Add m_Trav_Opt, "TRAV_OPT"

'Calculation for Seamless Connections

$m_Seamless = (m_rst("Q6A") * 50) + (m_rst("Q6B") * 50)$

```

    m_Values.Add m_Seamless, "SEAMLESS"
'Calculation for Safety
  If m_Safety_Ben > 0 Then
    m_Safety = 50 + (m_rst("Q7B") * 50)
  Else
    m_Safety = m_rst("Q7B") * 50
  End If
  m_Values.Add m_Safety, "O_SAFETY"
'Calculation for Security
  m_Security = 100 * m_rst("Q8A")
  m_Values.Add m_Security, "SECURITY"
'Calculation for Community Based Design
  'The -1 Multiplication Corrects the negative sign introduced for true
  m_Commnty = (-1) * ((20 * m_rst("Q9A")) + (20 * m_rst("Q9B")) + _
    (20 * m_rst("Q9C")) + (20 * m_rst("Q9D")) + _
    (20 * m_rst("Q9E")))
  m_Values.Add m_Commnty, "COMMNTY"
'Calculation for Collaborative Decision Making
  If m_rst("Q10B") < 5 Then
    m_Collab = (m_rst("Q10A") * 50) + (m_rst("Q10B") * 10)
  Else
    m_Collab = (m_rst("Q10A") * 50) + 50
  End If
  m_Values.Add m_Collab, "COLLAB"
'Calculation for Freight
  m_Freight = (75 * m_rst("Q11A")) + (25 * m_rst("Q11C"))
  m_Values.Add m_Freight, "FREIGHT"
'Calculation for Economic Prosperity
  m_Econ_Pros = (50 * m_rst("Q12A")) + (50 * m_rst("Q12B"))
  m_Values.Add m_Econ_Pros, "ECON_PROS"
'Calculation for Tourism
  m_Tourism = (50 * m_rst("Q13A")) + (50 * m_rst("Q13B"))
  m_Values.Add m_Tourism, "TOURISM"
'Calculation for the Air Quality
  If m_rst("Q14B") = 1 Then
    m_Air_Qual = 50 + m_rst("Q14A") * 50
  Else
    m_Air_Qual = 0
  End If
  m_Values.Add m_Air_Qual, "AIR_QUAL"
'Calculation for the Water Quality
  m_Wtr_Qual = (34 * m_rst("Q15A")) + (33 * m_rst("Q15B")) + _
    (33 * m_rst("Q15C"))
  m_Values.Add m_Wtr_Qual, "WTR_QUAL"
'Calculation for the Habitat
  m_Habitat = ((25 * m_rst("Q16A")) + (25 * m_rst("Q16B")) + _
    (25 * m_rst("Q16C")) + (25 * m_rst("Q16D"))) / _
    (1 + m_rst("Q16E") + m_rst("Q16F"))
  m_Values.Add m_Habitat, "HABITAT"
'Calculation for the Use of Resources
  m_Resource = 100 * m_rst("Q17")
  m_Values.Add m_Resource, "RESOURCE"

```

End Sub

Public Function GetItemValue(itemname As String) As Variant

```
Dim rtnval

rtnval = m_Values.Retrieve(itemname)

GetItemValue = rtnval
End Function

Private Sub Class_Terminate()
    a_rst.Close
    m_rst.Close
    m_dbs.Close
    Set a_rst = Nothing
    Set m_rst = Nothing
    Set m_dbs = Nothing
    Set m_Values = Nothing
End Sub
```

Program Code for Scenarios and Optimization

Building Scenario

Option Compare Database

Public Function BuildScenarioProjects() As Boolean

```

Dim qryDef As DAO.QueryDef
Dim t_rst As DAO.Recordset
Dim n_rst As DAO.Recordset
Dim strFilter As String
Dim PrjID As Integer
Dim TypID As Integer
Dim tmp As Boolean
Dim newColl As Collection

Set m_dbs = CurrentDb

' Open QueryDef object with one parameters.
Set qryDef = m_dbs.QueryDefs("scen_AllProjects")

' Set recordset to all Projects.
Set t_rst = qryDef.OpenRecordset
' Set recordset to all Projects.
Set n_rst = m_dbs.OpenRecordset("Project_Results", dbOpenDynaset)
'If there are projects in the Project_Results table delete them since
' calcs must be performed each time
While Not n_rst.EOF
    n_rst.Delete
    n_rst.MoveNext
Wend
' If it is a unique data entry then calculate all scores
If Not t_rst.EOF Then
    'Populate the record count property
    t_rst.MoveLast
    t_rst.MoveFirst
    'Open up the Progress Bar
    DoCmd.OpenForm "scen_dispProgress", acNormal, , , , t_rst.RecordCount

    'Alert user that this will take a while
    MsgBox "There are " & t_rst.RecordCount & " records to calculate, please be patient" & vbCrLf & _
        "All of the projects where the input is complete will be shown below."

    'Loop through all projects & their calculations
    While Not t_rst.EOF
        TypID = t_rst("Type_ID")
        Set objCalcs = PTypeCalcs(TypID)
        If Not objCalcs Is Nothing Then
            PrjID = t_rst("Project_ID")
            'Perform the calculations for the project and return status
            tmp = objCalcs.SetProjectNumber(PrjID)
            'If a projects input is done then add those results to the Project_Results table
            If objCalcs.IsInputComplete Then
                Set newColl = objCalcs.GetResultValues
                Add2Recordset n_rst, newColl, PrjID
                Set newColl = Nothing
            End If
        End If
    End While
    Set objCalcs = Nothing

```



```

        t_rst.MoveNext
        'Update progress bar on activities
        Forms!scen_dispProgress.IncrementBar
    Wend
    'Hide the progress bar - we're done
    DoCmd.Close acForm, "scen_dispProgress"
End If
t_rst.Close
n_rst.Close
qryDef.Close

```

```

Set t_rst = Nothing
Set n_rst = Nothing
Set qryDef = Nothing
Set m_dbs = Nothing

```

End Function

Private Sub Add2Recordset(tmprst As DAO.Recordset, coll As Collection, prjct_id As Integer)

```

'Add a new record to the Project_Results table
With tmprst
    .AddNew
        IP_ID = prjct_id
        IBCR = coll.item("BCR")
        IWSDOT_Cost = coll.item("WSDOT_COST")
        ISAFE_BEN = coll.item("SAFETY_BEN")
        ITT_BEN = coll.item("TT_BEN")
        IUSER_BEN = coll.item("USER_BEN")
        IEnv_Ben = coll.item("ENV_BEN")
        ISYS_OM = coll.item("SYS_OM")
        ISYS_PRES = coll.item("SYS_PRES")
        ISP_NEEDS = coll.item("SP_NEEDS")
        ICONG_REL = coll.item("CONG_REL")
        ITRAV_OPT = coll.item("TRAV_OPT")
        ISEAMLESS = coll.item("SEAMLESS")
        ISAFETY = coll.item("O_SAFETY")
        ISecurity = coll.item("SECURITY")
        ICOMMNTY = coll.item("COMMNTY")
        ICOLLAB = coll.item("COLLAB")
        IFREIGHT = coll.item("FREIGHT")
        IECON_PROS = coll.item("ECON_PROS")
        ITourism = coll.item("TOURISM")
        IAIR_QUAL = coll.item("AIR_QUAL")
        IWTR_QUAL = coll.item("WTR_QUAL")
        IHABITAT = coll.item("HABITAT")
        IResource = coll.item("RESOURCE")
    .Update
End With

```

End Sub

Private Function PTypeCalcs(pType As Integer) As Object

```

Select Case pType
    Case 19
        Set PTypeCalcs = New fry_calcConstruction_Terminal
    Case 7
        Set PTypeCalcs = New fry_calcConstruction_Vessel
    Case 5
        Set PTypeCalcs = New fry_calcPreservation_Vessel
    Case 9
        Set PTypeCalcs = New hwy_calcPreservation_Pavements
    Case 20

```

```

    Set PTypeCalcs = New its_calcIDAS
Case 21
    Set PTypeCalcs = New its_calcSCRITS
Case 16, 17, 18, 26
    Set PTypeCalcs = New nonmot_calcAll
Case 24
    Set PTypeCalcs = New tran_calcSPASM
Case 25
    Set PTypeCalcs = New tran_calcSTEAM
Case Else
    Set PTypeCalcs = Nothing
End Select
End Function

```

Optimize Scenario

Option Compare Database

```
Public Sub OptimizeRecords(wgtdata() As Double, L_budget As Long)
```

```

    Dim tCnt As Integer
    Dim rstTmp As DAO.Recordset
    Dim sumdata(22) As Double
    Dim prjsumScore(22) As Double

```

```
Set rstTmp = CurrentDb.OpenRecordset("scen_Selected")
```

```
If rstTmp.EOF Then Exit Sub
```

```

rstTmp.MoveLast
rstTmp.MoveFirst
rCount = rstTmp.RecordCount
alldata = rstTmp.GetRows(rCount)

```

```

' Calls the LINGO DLL callback function
'nErrorCode = LSsetCallback(nReserved, _
'AddressOf MyCallback, ByVal 0)

```

```

' This global variable will store the
' objective of the best solution so far
'dBestlp = 1E+30

```

```

' This array will be used to transfer data
' to and from LINGO
Dim dTransferArea() As Double

```

```

'Set up array's dimensions
dTcount = (24 * rCount) + 25
ReDim dTransferArea(dTcount)

```

```

'Set pointer #1 (record count)
tCnt = 1
dTransferArea(tCnt) = rCount

```

```

'Set pointer #2 (WSDOT costs)
tCnt = 2
For i = 0 To rCount - 1
    dTransferArea(tCnt) = alldata(1, i)
    tCnt = tCnt + 1
Next

```

```

'Set pointer #3 (Weighted Scores)
tCnt = 2 + rCount

```

```

For i = 0 To rCount - 1
  For j = 4 To UBound(alldata, 1)
    wghtdVal = alldata(j, i) * wgtdata(j - 3)
    sumdata(j - 3) = sumdata(j - 3) + wghtdVal
    dTransferArea(tCnt) = wghtdVal
    'Add up sum of all weighted scores per project
    prjsumScore(i) = prjsumScore(i) + wghtdVal
    tCnt = tCnt + 1
  Next
Next

'Set pointer #4 (Lump Sum Budget)
tCnt = 2 + (23 * rCount)
dTransferArea(tCnt) = L_budget

'Set pointer #5 (Goals)
tCnt = 3 + (23 * rCount)
For i = 1 To 22
  dTransferArea(tCnt) = sumdata(i)
  tCnt = tCnt + 1
Next

' Point to where each of the transfer areas
' begin in the transfer array
Dim nTransferPointers(7) As Long
nTransferPointers(1) = 1           'Number of Projects
nTransferPointers(2) = 2           'WSDOT costs
nTransferPointers(3) = 2 + rCount  'Weighted Scores
nTransferPointers(4) = 2 + (23 * rCount) 'Lump Sum Budget
nTransferPointers(5) = 3 + (23 * rCount) 'Goals for 22 Areas
nTransferPointers(6) = 25 + (23 * rCount) 'Fund Variable
nTransferPointers(7) = dTcount     'Status

' Build LINGO's command script (commands
' are terminated with an ASCII 10
Dim cScript As String

' Suppress solution reports
cScript = "SET TERSEO 1" & Chr(10)

' Read in the model file
cScript = cScript & _
"TAKE C:\LINGO6\SAMPLES\MICATEST2.LNG" & Chr(10)

' Solve the model
cScript = cScript & "GO" & Chr(10)

' Quit LINGO DLL
cScript = cScript & "QUIT" & Chr(10)

' Mark end of script with a null byte
cScript = cScript & Chr(0)

' Name of file to route standard output to
Dim cLogFile As String
cLogFile = "C:\LINGO6\SAMPLES\MICATEST.LOG"

' Run the LINGO script
Call LGVBSCRIPT(cScript, cLogFile, 7, _
dTransferArea(1), nTransferPointers(1), _
nErrorCode)

```

```
' Problems?
  If nErrorCode > 0 Or _
    dTransferArea(dTcount) > 0 Then
    i = MsgBox("Unable to solve!")
  End If

' Lingo is done - Do something
rCnt = 0
fCnt = 0
' Place Start values in dialog box
rstTmp.MoveFirst
tCnt = 25 + (23 * rCount)
With rstTmp
  While Not .EOF
    fVal = dTransferArea(tCnt)
    .Edit
    !Fund = fVal
    !Total_Score = prjsumScore(rCnt) / 100000
    .Update
    .MoveNext
    If fVal > 0 Then
      fCnt = fCnt + 1
    End If
    tCnt = tCnt + 1
    rCnt = rCnt + 1
  Wend
End With
rstTmp.Close
Set rstTmp = Nothing

  MsgBox CStr(fCnt) & " Projects Funded"
End Sub
```