```
                End If
                ShowAnalysisSheet
            End Select
          End If
        Case SECONDARY_TWO
'         --process selected cbg's
          If Not (SetTargetArray(Targets()) > 0) Then
            MsgBox "Select one or more items from list"
            Exit Sub
          Else
            Select Case Switch
              Case PRINT_INDIVIDUAL
                For index = 1 To UBound(Targets)
                  SingleTarget(1) = Targets(index)
                  RollupFile StateFile, SingleTarget, 6
                  MoveDataToReports tbPrimary.Text, CStr(Targets(index))
                  PrintReports
                Next index
              Case PRINT_SUMMARY
                RollupFile StateFile, Targets, 6
                If UBound(Targets) > 1 Then
                  MoveDataToReports tbPrimary.Text, "Multiple CBG's [" & UBound(Targ
ets) & "]"
                Else
                  MoveDataToReports tbPrimary.Text, "[" & Targets(1) & "]"
                End If
                PrintReports
              Case Else
                RollupFile StateFile, Targets, 6
                If UBound(Targets) > 1 Then
                  MoveDataToReports tbPrimary.Text, "Multiple CBG's [" & UBound(Targ
ets) & "]"
                Else
                  MoveDataToReports tbPrimary.Text, "[" & Targets(1) & "]"
                End If
                ShowAnalysisSheet
            End Select
          End If
        Case Else
'         --process entire state
          SaveDetail = (ThisWorkbook.Sheets("Options").CheckBoxes("AggregateSupportD
etail").Value = xlOn)
          RollupFile StateFile
          MoveDataToReports tbPrimary.Text, "Total State"
          If Switch <> REVIEW_SUMMARY Then
            PrintReports
            If SaveReports Then
              Set FoundCell = ThisWorkbook.Worksheets("RangeNames").Range("StateList
").Find(what:=tbPrimary.Text, lookat:=xlWhole, matchcase:=False)
              SaveReportWorkbook FoundCell.Offset(0, -1).Cells(1, 1).Text
            End If
          Else
            ShowAnalysisSheet
          End If
      End Select
  End Select

  ShowStatusMessage

End Sub
```

```
Public Sub SaveReportWorkbook(Directory As String)
Dim SaveFileName          As String
Dim wb                    As Workbook, _
    SaveWorkbook          As Workbook


  Application.DisplayAlerts = False

  SaveFileName = ThisWorkbook.Path & "\reports\report.xls"

' --get reference to report workbook
  For Each wb In Workbooks
    If wb.FullName = SaveFileName Then
      Set SaveWorkbook = wb
    End If
  Next wb

  SaveFileName = ThisWorkbook.Path & "\reports\" & Directory & "\" & ThisWorkbook.Sh
eets("RangeNames").Range("SavedReportName")

  ShowStatusMessage "saving " & SaveFileName

  With SaveWorkbook
    .SaveAs FileName:=SaveFileName, FileFormat:=xlNormal, _
            Password:="", WriteResPassword:="", ReadOnlyRecommended:=False, _
            CreateBackup:=False
    .Close
  End With

  Set wb = Nothing
  Set SaveWorkbook = Nothing

  Application.DisplayAlerts = True


End Sub

Public Function SetStateString() As String
Dim FoundCell          As Range
Dim index              As Long
Dim rtn                As String

' --if there are more than 20 states, return only number processed
  If UBound(StatesUsed) > 20 Then
    SetStateString = "Multiple States [" & UBound(StatesUsed) & "]"
    Exit Function
  End If

' --if only one state, return state name
  If UBound(StatesUsed) = 1 Then
    Set FoundCell = ThisWorkbook.Worksheets("RangeNames").Range("StateAbbreviationLi
st").Find(what:=StatesUsed(1), lookat:=xlWhole)
    SetStateString = FoundCell.Offset(0, 1).Cells(1, 1).Text
    Exit Function
  End If

' --build string of state abbreviations
  rtn = StatesUsed(1)

  For index = 2 To UBound(StatesUsed)
```

```
    rtn = rtn & ","
    rtn = rtn & StatesUsed(index)
  Next index

  SetStateString = "[" & rtn & "]"

End Function

Public Function CheckReportSelections() As Boolean
Dim dd                  As DropDown
Dim tbPrimary           As Textbox

  Set tbPrimary = ThisWorkbook.Sheets("Reports").TextBoxes("txtPrimaryKey")

' --make sure a model is defined
  If CurrentModel.Name = "" Then
    Set dd = ThisWorkbook.Sheets("Reports").DropDowns("ddModels")
    SetCurrentModel dd.List(dd.ListIndex)
    Set dd = Nothing
  End If

  If tbPrimary.Text = "" Then
    MsgBox "Must select a Primary Key"
    CheckReportSelections = False
    Exit Function
  End If

  Set dd = ThisWorkbook.Sheets("Reports").DropDowns("FactorSets")
  If dd.ListIndex < 1 Then
    MsgBox "Must select a Factor Set"
    CheckReportSelections = False
    Exit Function
  End If

  CheckReportSelections = True

  Set dd = Nothing
  Set tbPrimary = Nothing

End Function

Public Function SetTargetArray(Array() As Variant) As Long
Dim index                 As Long, _
    TargetIndex           As Long
Dim lst                   As Listbox

  Set lst = ThisWorkbook.Sheets("Reports").ListBoxes("lstOptions")

' --loop through all items in list
  For index = 1 To lst.ListCount
    If lst.Selected(index) Then
      TargetIndex = TargetIndex + 1
      ReDim Preserve Array(TargetIndex)
      Array(TargetIndex) = lst.List(index)
    End If
  Next index

  SetTargetArray = TargetIndex

End Function
```

```
Public Function CheckSelectedStateFiles(Array() As String) As Long
Dim tbPrimary              As Textbox
Dim index                  As Long, _
    FileIndex              As Long, _
    dotPosition            As Long, _
    RecordPosition         As Long
Dim lst                    As Listbox
Dim StateRng               As Range
Dim Cosa                   As String, _
    FileName               As String, _
    FilePath               As String, _
    IndexFile              As String, _
    target                 As String
Dim UseFile                As Boolean

  Set StateRng = ThisWorkbook.Sheets("RangeNames").Range("StateList")
  Set lst = ThisWorkbook.Sheets("Reports").ListBoxes("lstOptions")
  Set tbPrimary = ThisWorkbook.Sheets("Reports").TextBoxes("txtPrimaryKey")
' --initialize variables
  FilePath = ThisWorkbook.Path & "\results\"
  target = tbPrimary.Text

' --loop through all states checking for files
  For index = 1 To lst.ListCount
    If lst.Selected(index) Then
      Cosa = StateRng.Offset(index - 1, -1).Cells(1, 1).Text
      FileName = FilePath & Cosa & "\" & CurrentModel.Name & ".csv"
'     --assume file won't be used
      UseFile = False
      If FileExists(FileName) Then
'       --define index file
        dotPosition = InStr(FileName, ".")
        IndexFile = Left$(FileName, dotPosition) & "idx"
'       --need to check for "Holding Company", "Company"
        Select Case PrimaryKey
          Case PRIMARY_HOLDINGCOMPANY
'           --RecordPosition is -1 if not found
            RecordPosition = GetIniLong(target, "Holding Companies", IndexFile)
            UseFile = RecordPosition > 0
          Case PRIMARY_COMPANY
'           --RecordPosition is -1 if not found
            RecordPosition = GetIniLong(target, "Companies", IndexFile)
            UseFile = RecordPosition > 0
          Case Else
            UseFile = True
        End Select
      End If
      If UseFile Then
        FileIndex = FileIndex + 1
        ReDim Preserve Array(FileIndex)
        Array(FileIndex) = FileName
      Else
        lst.Selected(index) = False
      End If
    End If
  Next index

  CheckSelectedStateFiles = FileIndex

End Function
```

```
Public Sub CheckAllStateFiles(Array() As String)
Dim index                 As Long
Dim StateRng              As Range, _
    Cell                  As Range
Dim Cosa                  As String, _
    FileName              As String, _
    FilePath              As String

  Set StateRng = ThisWorkbook.Sheets("RangeNames").Range("StateList")
  FilePath = ThisWorkbook.Path & "\results\"

' --loop through all states checking for files
  For Each Cell In StateRng.Cells
    Cosa = Cell.Offset(0, -1).Cells(1, 1).Value
    FileName = FilePath & Cosa & "\" & CurrentModel.Name & ".csv"
    If FileExists(FileName) Then
      index = index + 1
      ReDim Preserve Array(index)
      Array(index) = FileName
    End If
  Next Cell

End Sub

Public Sub CopyReportInformation(Description As String, ReportType As String, InfFil
e As String)
Dim sht                 As Worksheet

  Set sht = ActiveWorkbook.Worksheets("Variables")

  With sht
    .Activate
    .Range("InputData").Value = UCase(CurrentModel.Name) & ".CSV, " & InfFile
    .Range("WorkbookName").Value = CurrentModel.LogicSet
    .Range("Description").Value = CurrentModel.Description
    .Range("State").Value = Description
    .Range("ReportInd").Value = ReportType
  End With

  Set sht = Nothing

End Sub

Public Sub CopyExpenseData(IniFile As String)
Dim sht                 As Worksheet
Dim TableRng            As Range, _
    Cell                As Range
Dim index               As Long
Dim RtnString           As String, _
    target              As String
Dim Record              As Variant, _
    Records()           As Variant

  Set sht = ActiveWorkbook.Worksheets("Expense Inputs")
  sht.Activate

' --load Aggregate Support
  Set TableRng = sht.Range("AggregateSupportTable")
  For index = 1 To 7
    RtnString = GetIniString(IniFile, "Aggregate Support", CStr(index))
    TableRng.Offset(index - 1, 0).Cells(1, 1).Value = CDbl(RtnString)
```

```
  Next index

' --load Common Loadings
  Set TableRng = sht.Range("CommonLoadings")
  For Each Cell In TableRng.columns(1).Cells
    target = Cell.Cells(1, 1).Text
    RtnString = GetIniString(IniFile, "Common Loadings", target)
    Cell.Offset(0, 1).Cells(1, 1).Value = CDbl(RtnString)
  Next Cell

' --load Support Ratio Table
  Set TableRng = sht.Range("SupportRatioTable")
  For Each Cell In TableRng.columns(1).Cells
    index = index + 1
    target = Cell.Cells(1, 1).Text
    RtnString = GetIniString(IniFile, "Support Ratio Table", target)
    Record = ParseRecord(RtnString, 3)
    Cell.Offset(0, 1).Resize(1, 3).Value = Record
  Next Cell

' --load Monthly Operating Expenses
  Set TableRng = sht.Range("MonthlyOperatingExpenses")
  For Each Cell In TableRng.columns(1).Cells
    index = index + 1
    target = Cell.Cells(1, 1).Text
    RtnString = GetIniString(IniFile, "Monthly Operating Expenses - Residential", ta
rget)
    Record = ParseRecord(RtnString, 3)
    Cell.Offset(0, 2).Resize(1, 3).Value = Record
    RtnString = GetIniString(IniFile, "Monthly Operating Expenses - Business", targe
t)
    Record = ParseRecord(RtnString, 3)
    Cell.Offset(0, 5).Resize(1, 3).Value = Record
  Next Cell

  Set sht = Nothing
  Set Cell = Nothing
  Set TableRng = Nothing

End Sub

Public Sub CopyCapitalCostData(IniFile As String)
Dim NameRng          As Range, _
    TableRng         As Range, _
    Cell             As Range
Dim sht              As Worksheet
Dim AccountName      As String, _
    IniData          As String
Dim Records()        As Variant
Dim index            As Long

  ReDim Records(19)

  Set sht = ActiveWorkbook.Worksheets("Capital Cost Inputs")
  sht.Activate

  Set NameRng = sht.Range("ACFAccountNames")
  Set TableRng = sht.Range("ACFOutputData")

  For Each Cell In NameRng.columns(1).Cells
    index = index + 1
```

```
    AccountName = Cell.Cells(1, 1).Text
    IniData = GetIniString(IniFile, "Account Capital Cost", AccountName)
    Records(index) = ParseRecord(IniData, 7)
  Next Cell

  TableRng.Value = Records

  Set NameRng = Nothing
  Set TableRng = Nothing
  Set sht = Nothing

End Sub

Public Sub CopyReportData()
Dim stream            As Long, _
    lineIndex         As Long, _
    columns           As Long
Dim Start             As Range
Dim ReportFile        As String, _
    buffer            As String
Dim Record            As Variant
Dim sht               As Worksheet

  Set sht = ActiveWorkbook.Worksheets("Report Inputs")
  sht.Activate

  ReportFile = ThisWorkbook.Path & "\reports\rptData.csv"

  Set Start = ActiveWorkbook.Worksheets("Report Inputs").Range("InputStartLocation")

  stream = FreeFile()
  Open ReportFile For Input Shared As #stream

  lineIndex = -1

  Do Until EOF(stream)
    Line Input #stream, buffer
    lineIndex = lineIndex + 1
    If Len(buffer) Then
      If Len(buffer) > 100 Then
        Record = ParseRecord(buffer, 60)
        columns = UBound(Record)
      Else
        Record = buffer
        columns = 1
      End If
      Start.Offset(lineIndex, 0).Resize(1, columns).Value = Record
    End If
  Loop

  Close stream

End Sub

Public Function SetStateFile() As String
Dim FoundCell     As Range
Dim Cosa          As String, _
    ResultsPath   As String, _
    StateName     As String

  StateName = ThisWorkbook.Worksheets("Reports").TextBoxes("txtPrimaryKey").Text
```

```
  Set FoundCell = ThisWorkbook.Worksheets("RangeNames").Range("StateList").Find(what
:=StateName, lookat:=xlWhole)
  Cosa = FoundCell.Offset(0, -1).Cells(1, 1).Text

  ResultsPath = ThisWorkbook.Path & "\results\"

  SetStateFile = ResultsPath & Cosa & "\" & CurrentModel.Name & ".csv"

End Function

Public Function FullStateName(FileName As String) As String
Dim FoundCell     As Range
Dim Cosa          As String, _
    FilePath      As String

  FilePath = ExtractPath(FileName)
  Cosa = Left$(Right$(FilePath, 3), 2)

  Set FoundCell = ThisWorkbook.Worksheets("RangeNames").Range("StateAbbreviationList
").Find(what:=Cosa, lookat:=xlWhole)

  FullStateName = FoundCell.Offset(0, 1).Cells(1, 1).Text

End Function

Public Sub SetPrimaryDisplay()
Dim List          As Listbox
Dim sht           As Worksheet

  Set sht = ThisWorkbook.Sheets("Reports")
  Set List = sht.ListBoxes("lstOptions")

  sht.Unprotect

  List.RemoveAllItems
  sht.TextBoxes("txtPrimaryKey").Text = ""
  SecondaryKey = 0
  LastOption = PRIMARY

' --set things up for National
  If sht.OptionButtons("National").Value = xlOn Then
    With sht.OptionButtons("Option 1")
      .Visible = False
      .Value = xlOff
    End With
    With sht.OptionButtons("Option 2")
      .Visible = False
      .Value = xlOff
    End With
    sht.TextBoxes("txtPrimaryKey").Text = "National"
    PrimaryKey = PRIMARY_NATIONAL
  End If

' --setup for Holding Company
  If sht.OptionButtons("Holding Company").Value = xlOn Then
    With sht.OptionButtons("Option 1")
      .Caption = "Total"
      .Value = xlOff
      .Visible = True
    End With
```

```
    With sht.OptionButtons("Option 2")
      .Caption = "State(s)"
      .Value = xlOff
      .Visible = True
    End With
    LoadHoldingCompanies List
    PrimaryKey = PRIMARY_HOLDINGCOMPANY
  End If

' --setup for Company
  If sht.OptionButtons("Company").Value = xlOn Then
    With sht.OptionButtons("Option 1")
      .Caption = "Total"
      .Visible = True
      .Value = xlOff
    End With
    With sht.OptionButtons("Option 2")
      .Caption = "State(s)"
      .Value = xlOff
      .Visible = True
    End With
    LoadCompanies List
    PrimaryKey = PRIMARY_COMPANY
  End If

' --setup for State
  If sht.OptionButtons("State").Value = xlOn Then
    With sht.OptionButtons("Option 1")
      .Caption = "Wire Center(s)"
      .Value = xlOff
      .Visible = True
    End With
'   --need to check for "cbg" or "wire center processing"
    If ThisWorkbook.Sheets("Options").OptionButtons("AggregateSupportWC").Value = xl
On Then
      With sht.OptionButtons("Option 2")
        .Caption = ""
        .Value = xlOff
        .Visible = False
      End With
    Else
      With sht.OptionButtons("Option 2")
        .Caption = "Census Block Group(s)"
        .Value = xlOff
        .Visible = True
      End With
    End If
    LoadStates List
    PrimaryKey = PRIMARY_STATE
  End If

  sht.Protect

  Set sht = Nothing
  Set List = Nothing

End Sub

Public Sub SetSecondaryDisplay()
Dim List          As Listbox
Dim sht           As Worksheet
```

```
Dim dd              As DropDown
Dim tb              As Textbox


' --make sure a model is defined
  If CurrentModel.Name = "" Then
    Set dd = ThisWorkbook.Sheets("Reports").DropDowns("ddModels")
    On Error Resume Next
    SetCurrentModel dd.List(dd.ListIndex)
    If Err <> 0 Then
      MsgBox "Must select Model first"
      Err = 0
      Exit Sub
    End If
    On Error GoTo 0
    Set dd = Nothing
  End If

  Set sht = ThisWorkbook.Sheets("Reports")
  Set List = sht.ListBoxes("lstOptions")
  Set tb = sht.TextBoxes("txtPrimaryKey")

  LastOption = SECONDARY

  If sht.OptionButtons("Option 1").Value = xlOn Then
    SecondaryKey = SECONDARY_ONE
    Select Case PrimaryKey
      Case PRIMARY_STATE
        If Len(tb.Text) Then
          SetPaths tb.Text
          LoadWireCenters List
        Else
          MsgBox "Must select State as Primary Key"
          sht.OptionButtons("Option 1").Value = xlOff
          LastOption = PRIMARY
        End If
      Case Else
        List.RemoveAllItems
    End Select
  End If

  If sht.OptionButtons("Option 2").Value = xlOn Then
    SecondaryKey = SECONDARY_TWO
    Select Case PrimaryKey
      Case PRIMARY_HOLDINGCOMPANY
        LoadStates List
      Case PRIMARY_COMPANY
        LoadStates List
      Case PRIMARY_STATE
        If Len(tb.Text) Then
          SetPaths tb.Text
          LoadCensusBlockGroups List
        Else
          MsgBox "Must select State as Primary Key"
          sht.OptionButtons("Option 2").Value = xlOff
          LastOption = PRIMARY
        End If
    End Select
  End If

  Set List = Nothing
```

```
  Set sht = Nothing
  Set tb = Nothing

End Sub

Public Sub ClearReportsSheet()
Dim obj              As Object
Dim sht              As Worksheet

  Set sht = ThisWorkbook.Worksheets("Reports")

  For Each obj In sht.TextBoxes
    obj.Text = ""
  Next obj

  For Each obj In sht.DropDowns
    obj.ListIndex = 0
  Next obj

  For Each obj In sht.OptionButtons
    obj.Value = xlOff
  Next obj

  sht.OptionButtons("Option 1").Visible = False
  sht.OptionButtons("Option 2").Visible = False

  For Each obj In sht.ListBoxes
    obj.RemoveAllItems
  Next obj

  PrimaryKey = 0
  SecondaryKey = 0

End Sub

Public Sub SelectAll()
Dim List           As Listbox
Dim Choices()      As Boolean
Dim index          As Long

  Set List = ThisWorkbook.Sheets("Reports").ListBoxes("lstOptions")

  If List.ListCount Then
    ReDim Choices(List.ListCount)

    For index = 1 To List.ListCount
      Choices(index) = xlOn
    Next index

    List.Selected = Choices
  End If

End Sub

Public Sub LoadCensusBlockGroups(lstAny As Listbox)
Dim stream               As Long, _
    index                As Long
Dim WireCenter           As String, _
    buffer               As String, _
    FileName             As String
Dim CensusBlocks()       As Variant
```

```
Dim Record                As Variant

    lstAny.RemoveAllItems

    FileName = ThisWorkbook.Sheets("RangeNames").Range("Input_Path") & CurrentModel.
CBGDataSet

    If FileExists(FileName) Then
'     --open input file
      stream = FreeFile()
      Open FileName For Input Shared As #stream

      ShowStatusMessage "Loading Census Block Groups ...."

      Do Until EOF(stream)
        Line Input #stream, buffer
        index = index + 1
        Record = ParseRecord(buffer)
        ReDim Preserve CensusBlocks(index)
        CensusBlocks(index) = Record(6)
      Loop

      ShowStatusMessage "sorting census block groups"
      SelectionSort CensusBlocks

      lstAny.List = CensusBlocks

      Close stream

      ShowStatusMessage
    Else
      ShowStatusMessage "Error: " & FileName & " is missing"
    End If

End Sub

Public Sub SelectionSort(Array As Variant)
Dim i                   As Long, _
    j                   As Long, _
    min                 As Long, _
    max                 As Long
Dim best_value          As Variant, _
    best_j              As Variant

' --this is a selection sort, not very quick, but easy to understand
' --keys are variants because cbg #'s are 14 digits long

  min = 1
  max = UBound(Array)

  For i = min To max - 1
    best_value = Array(i)
    best_j = i
    For j = i + 1 To max
      If Array(j) < best_value Then
        best_value = Array(j)
        best_j = j
      End If
    Next j
    Array(best_j) = Array(i)
    Array(i) = best_value
```

```
  Next i

End Sub

Public Sub SavePrimarySelection()
Dim List          As Listbox
Dim sht           As Worksheet
Dim index         As Long

  If LastOption <> PRIMARY Then Exit Sub

  Set sht = ThisWorkbook.Sheets("Reports")
  Set List = sht.ListBoxes("lstOptions")

  For index = 1 To List.ListCount
    If List.Selected(index) Then
      sht.TextBoxes("txtPrimaryKey").Text = List.List(index)
      Exit Sub
    End If
  Next index

  Set List = Nothing
  Set sht = Nothing

End Sub

Private Sub LoadStates(anyList As Listbox)
Dim rng             As Range, _
    Cell            As Range
Dim States()        As Variant
Dim index           As Long

  anyList.RemoveAllItems

  Set rng = ThisWorkbook.Sheets("RangeNames").Range("StateList")

  ReDim States(rng.rows.Count)

  For Each Cell In rng.columns(1).Cells
      index = index + 1
      States(index) = UCase$(Cell.Cells(1, 1).Text)
  Next Cell

  anyList.List = States

End Sub

Private Sub LoadCompanies(anyList As Listbox)
Dim stream        As Long, _
    index         As Long
Dim Items()       As String, _
    buffer        As String

  anyList.RemoveAllItems

  stream = FreeFile()
  Open ThisWorkbook.Path & "\company.txt" For Input Shared As #stream

  Do Until EOF(stream)
    Line Input #stream, buffer
    If Len(buffer) Then
```

```
      index = index + 1
      ReDim Preserve Items(index)
      Items(index) = buffer
    End If
  Loop

  Close stream

  anyList.List = Items

End Sub

Private Sub LoadHoldingCompanies(anyList As Listbox)
Dim stream        As Long, _
    index         As Long
Dim Items()       As String, _
    buffer        As String

  anyList.RemoveAllItems

  stream = FreeFile()
  Open ThisWorkbook.Path & "\parent.txt" For Input Shared As #stream

  Do Until EOF(stream)
    Line Input #stream, buffer
    If Len(buffer) Then
      index = index + 1
      ReDim Preserve Items(index)
      Items(index) = buffer
    End If
  Loop

  Close stream

  anyList.List = Items

End Sub
```

```
Option Explicit: Option Base 1: Option Compare Text

' --constants
  Private Const NumberOfCompanySizes = 3
  Private Const NumberOfDensityGroups = 7

' --data members set in Class Initialize routine
  Private NumberOfAccounts        As Long

' --data members set via subroutines
  Private ModelIniFile            As String
  Private FactorIniFile           As String
  Private cbgRecord               As Variant
  Private wcRecord                As Variant
  Private CBGMonthlyCost          As Double
  Private ValidationRow           As Long
  Public ShowDetail               As Boolean
  Public SaveDetail               As Boolean

' --public data members
  Public FileName                 As String
  Public OutputFile               As String
  Public NumberOfErrors           As Long
  Public ErrorFile                As String
  Public FilePosition             As Long
  Public ExtractSpecific          As Boolean
  Public StatesUsed()             As String
  Public StateIndex               As Long

  Private CompanySizeLabels()     As String
  Private DensityGroupLabels()    As String

' --these arrays contain account information
  Private NumberOfInputFields     As Long
  Private NumberOfStringFields    As Long
  Private CapCostFactors()        As Double
  Private InputSource()           As Double
  Private HHByCost()              As Long
  Private HHBYLoop()              As Long
  Private minLoopLength           As Double
  Private maxLoopLength           As Double
  Private CumulativeLines         As Double
  Private SingleBusLineFctr       As Double
  Private SingleBusinessLines     As Double
  Private LoopCol                 As Long
  Private HHCol                   As Long
  Private LoopCostCol             As Long
  Private CBGLinesCol             As Long
  Private CompanySizeCol          As Long
  Private DensityCol              As Long
  Private BusLinesCol             As Long
  Private CSIndex                 As Long

' --this is an array of variable size
  Private BenchmarkDollars()      As Long
  Private BenchMarkSupport()      As Double
  Private Benchmarks              As Long
  Private Expenses(3)             As Double
  Private SupportFactors()        As Double
  Private Totals()                As Double
```

```
Private Function GetCompanySizeIndex(Record As Variant) As Long
Dim index        As Long

  Select Case Record(CompanySizeCol)
    Case "S"
      index = 1
    Case "M"
      index = 2
    Case "L"
      index = 3
    Case Else
      index = 0
  End Select

  GetCompanySizeIndex = index

End Function

Private Sub ApplySegmentSupport(Record As Variant, dstream As Long)
Dim Accnt              As Long, _
    index              As Long, _
    costIndex          As Long, _
    accntIndex         As Long
Dim TotalCost          As Double
Dim AccountCost        As Double
Dim SupportCost        As Double
Dim ws                 As Worksheet
Dim ColumnIndex        As Long


' --Uncapped index is 1, Capped index is 2

  Set ws = ThisWorkbook.Sheets("Aggregate Support Calculations")
  ReDim BenchMarkSupport(Benchmarks + 1, 2)

  CSIndex = GetCompanySizeIndex(Record)
  SingleBusinessLines = SingleBusinessLines + Record(BusLinesCol) * SingleBusLineFct
r

' --set as module variable in Initialize_Rollup
  If ShowDetail Then
    ws.Cells(ValidationRow, 1).Value = Record(1)
    ws.Cells(ValidationRow, 2).Value = Record(4)
    ws.Cells(ValidationRow, 3).Value = Record(2)
    ws.Cells(ValidationRow, 4).Value = Record(6)
  End If

' --column numbers taken form models.inf "[Output Columns"]"
  If SaveDetail Then
'   --write identifiers
    Write #dstream, Record(1), Record(4), Record(2), Record(6);
  End If

  For costIndex = 1 To 2
    CBGMonthlyCost = 0
    SupportCost = 0
    For Accnt = 1 To 19
      AccountCost = 0
      If Record(CBGLinesCol) <> 0 Then
```

```
        accntIndex = InputSource(Accnt, costIndex)
        AccountCost = CapCostFactors(Accnt) * SupportFactors(Accnt, CSIndex) * Recor
d(accntIndex) / Record(CBGLinesCol) / 12
        If SupportFactors(Accnt, CSIndex) <> 1 Then
          SupportCost = SupportCost + SupportFactors(Accnt, CSIndex) * Record(accntI
ndex)
        End If
        If ShowDetail Then
'         --this is column prior to beginning column
          ColumnIndex = (costIndex - 1) * (20 * 5 + 7) + (Accnt - 1) * 5 + 4
          ws.Cells(ValidationRow, ColumnIndex + 1).Value = CapCostFactors(Accnt)
          ws.Cells(ValidationRow, ColumnIndex + 2).Value = SupportFactors(Accnt, CSI
ndex)
          ws.Cells(ValidationRow, ColumnIndex + 3).Value = Record(accntIndex)
          ws.Cells(ValidationRow, ColumnIndex + 4).Value = Record(CBGLinesCol)
          ws.Cells(ValidationRow, ColumnIndex + 5).Value = AccountCost
'         --skips "Total" calculations last time through loop only, otherwise has no
 effect
          ColumnIndex = ColumnIndex + 6
        End If
      End If
      CBGMonthlyCost = CBGMonthlyCost + AccountCost
    Next Accnt

'   --only save capped data
    If SaveDetail Then
      If costIndex = 2 Then
'       --write "Investment per line data"
        Write #dstream, Record(20) / Record(CBGLinesCol), _
                    Record(23) / Record(CBGLinesCol), _
                    Record(24) / Record(CBGLinesCol), _
                    (SupportCost + Record(48) - Record(20) - Record(23) - Record(24)
) / Record(CBGLinesCol), _
                    (SupportCost + Record(48)) / Record(CBGLinesCol);
'       --write "Expense Per Month Data"
        Write #dstream, CBGMonthlyCost, Expenses(CSIndex), CBGMonthlyCost + Expenses
(CSIndex);
'       --write "Line Data"
        Write #dstream, Record(13), _
                        Record(HHCol), _
                        Record(CBGLinesCol) - Record(8), _
                        Record(BusLinesCol) * SingleBusLineFctr, _
                        Record(BusLinesCol) * (1 - SingleBusLineFctr), _
                        Record(CBGLinesCol);
      End If
    End If

'   --add per line expense to totalcost
    CBGMonthlyCost = CBGMonthlyCost + Expenses(CSIndex)

    If ShowDetail Then
      ColumnIndex = ColumnIndex + 2
      ws.Cells(ValidationRow, ColumnIndex).Value = Expenses(CSIndex)
      ColumnIndex = ColumnIndex + 1
      ws.Cells(ValidationRow, ColumnIndex).Value = (Record(HHCol) + Record(BusLinesC
ol) * SingleBusLineFctr)
      ColumnIndex = ColumnIndex + 2
    End If

'   --monthly costs are stored as first benchmark
```

```
    BenchMarkSupport(1, costIndex) = CBGMonthlyCost

    For index = 1 To Benchmarks
      --index really runs from 2 to Benchmarks+1
      If (CBGMonthlyCost - BenchmarkDollars(index)) < 0 Then
        BenchMarkSupport(index + 1, costIndex) = 0
      Else
        BenchMarkSupport(index + 1, costIndex) = (CBGMonthlyCost - BenchmarkDollars(
index)) * (Record(HHCol) + Record(BusLinesCol) * SingleBusLineFctr) * 12
      End If
      If SaveDetail Then
        If costIndex = 2 Then
          --write "Aggregate Support Data"
          Write #dstream, BenchMarkSupport(index + 1, costIndex);
        End If
      End If
      If ShowDetail Then
        ws.Cells(ValidationRow, ColumnIndex).Value = BenchMarkSupport(index + 1, cos
tIndex)
        ColumnIndex = ColumnIndex + 1
      End If
    Next index
  Next costIndex

  If ShowDetail Then ValidationRow = ValidationRow + 1
  If SaveDetail Then Write #dstream,

  Set ws = Nothing

End Sub

Public Sub FormatAggregateSupportSheet()
Dim rng1                    As Range, _
    rng2                    As Range, _
    rng3                    As Range, _
    rng4                    As Range
Dim NumberOfRows            As Long


  If ShowDetail Then
    --number of rows needs to be greater than 0, 4 is the first data row
    NumberOfRows = ValidationRow - 4
    Set rng1 = ThisWorkbook.Sheets("Aggregate Support Calculations").Range("ascformu
las1").Resize(rowsize:=NumberOfRows)
    Set rng2 = ThisWorkbook.Sheets("Aggregate Support Calculations").Range("ascformu
las2").Resize(rowsize:=NumberOfRows)
    Set rng3 = ThisWorkbook.Sheets("Aggregate Support Calculations").Range("ascformu
las3").Resize(rowsize:=NumberOfRows)
    Set rng4 = ThisWorkbook.Sheets("Aggregate Support Calculations").Range("ascformu
las4").Resize(rowsize:=NumberOfRows)
    If NumberOfRows > 1 Then
      rng1.FillDown
      rng2.FillDown
      rng3.FillDown
      rng4.FillDown
    End If
    --make sure column widths are ok
    Sheets("Aggregate Support Calculations").columns.AutoFit
    'Cells.Select
    'Range("S1").Activate
```

```
    'Selection.columns.AutoFit
    'Cells(1, 1).Activate
  End If

  Set rng1 = Nothing
  Set rng2 = Nothing
  Set rng3 = Nothing
  Set rng4 = Nothing

End Sub

Public Function SummarizeByWireCenter(FileName As String, Optional target As Variant
, Optional FieldNumber As Variant) As Boolean
Dim stream            As Long, _
    dstream           As Long, _
    index             As Long, _
    estream           As Long, _
    RecordPosition    As Long
Dim buffer            As String, _
    IdxFile           As String, _
    State             As String, _
    RtnString         As String, _
    HoldWC            As String, _
    DetailFile        As String
Dim Cost              As Double
Dim FindTarget        As Boolean, _
    DoWholeFile       As Boolean, _
    FirstRecord       As Boolean, _
    RecordPending     As Boolean
Dim tmpRecord         As Variant


  SummarizeByWireCenter = True

  If IsMissing(target) Then
    DoWholeFile = True
  End If

  If Not IsMissing(FieldNumber) Then
    index = InStr(FileName, ".")
    IdxFile = Left(FileName, index) & "idx"
    Select Case FieldNumber
      Case 1
        --wire center
        RecordPosition = GetIniLong(CStr(target), "Wire Centers", IdxFile)
        FindTarget = True
      Case 2
        --company
        RecordPosition = GetIniLong(CStr(target), "Companies", IdxFile)
        FindTarget = True
      Case 4
        --holding company
        RecordPosition = GetIniLong(CStr(target), "Holding Companies", IdxFile)
        FindTarget = True
      Case Else
        --cbg
        FindTarget = False
    End Select
    ShowStatusMessage "summarizing by wire center " & FileName & "   [" & LCase(targ
et) & "]"
```

```
  Else
    ShowStatusMessage "Summarizing by wire center " & FileName
  End If

• --set single business line factor
  State = FullStateName(FileName)
  RtnString = GetIniString(FactorIniFile, "Single Business Line Table", State)
  SingleBusLineFctr = CDbl(RtnString)

  stream = FreeFile()
  Open FileName For Input Shared As #stream

  estream = FreeFile()
  Open ErrorFile For Output As #estream

  dstream = FreeFile()
  If SaveDetail Then
•   --build root of file name
    DetailFile = Left$(FileName, InStr(FileName, ".")) & "wc"
    Open DetailFile For Output As #dstream
  End If

• --move to correct file position
  If FindTarget Then
    If RecordPosition > 0 Then
      Seek #stream, RecordPosition
    Else
      SummarizeByWireCenter = False
      Close stream, estream, dstream
      Exit Function
    End If
  End If

  FirstRecord = True
  RecordPending = False

  Do Until EOF(stream)
    Line Input #stream, buffer
    If Len(buffer) < 150 Then
      NumberOfErrors = NumberOfErrors + 1
      Print #estream, buffer
    Else
      tmpRecord = ParseRecord(buffer, 48)
      If DoWholeFile Then
•       --process every record, cumulating until wire center changes
        If Trim$(tmpRecord(1)) <> HoldWC Then
          If FirstRecord Then
            FirstRecord = False
          Else
•           --processing pending record
            ApplySegmentSupport wcRecord, dstream
            CumulateCBGData wcRecord
            CumulateHouseholds wcRecord
            MinMaxLoopLengths wcRecord
            CountHighCostLines wcRecord
            RecordPending = False
          End If
          HoldWC = Trim$(tmpRecord(1))
          wcRecord = tmpRecord
          RecordPending = True
```

```
        Else
          --cumulate if there is no change
          AddToWireCenterRecord tmpRecord
        End If
      Else
        --doing some kind of pattern search
        If FindTarget Then
          If Trim$(tmpRecord(FieldNumber)) = target Then
            If Trim$(tmpRecord(1)) <> HoldWC Then
              If FirstRecord Then
                FirstRecord = False
              Else
                --processing pending record
                ApplySegmentSupport wcRecord, dstream
                CumulateCBGData wcRecord
                CumulateHouseholds wcRecord
                MinMaxLoopLengths wcRecord
                CountHighCostLines wcRecord
                RecordPending = False
              End If
              HoldWC = Trim$(tmpRecord(1))
              wcRecord = tmpRecord
              RecordPending = True
            Else
              --cumulate if there is no change
              AddToWireCenterRecord tmpRecord
            End If
          Else
            --last matching record already processed
            Exit Do
          End If
        End If
      End If
    End If
  Loop

  --process last record
  If RecordPending Then
    ApplySegmentSupport wcRecord, dstream
    CumulateCBGData wcRecord
    CumulateHouseholds wcRecord
    MinMaxLoopLengths wcRecord
    CountHighCostLines wcRecord
  End If

  Close stream, estream, dstream

End Function

Public Function Summarize(FileName As String, Optional target As Variant, Optional F
ieldNumber As Variant) As Boolean
Dim stream            As Long, _
    dstream           As Long, _
    index             As Long, _
    estream           As Long, _
    RecordPosition    As Long
Dim buffer            As String, _
    IdxFile           As String, _
    State             As String, _
    RtnString         As String, _
```

```
    DetailFile          As String
Dim Cost                As Double
Dim FindTarget          As Boolean, _
    DoWholeFile         As Boolean, _
    FoundCBG            As Boolean

  Summarize = True

  If IsMissing(target) Then
    DoWholeFile = True
  End If

  If Not IsMissing(FieldNumber) Then
    index = InStr(FileName, ".")
    IdxFile = Left(FileName, index) & "idx"
    Select Case FieldNumber
      Case 1
'       --wire center
        RecordPosition = GetIniLong(CStr(target), "Wire Centers", IdxFile)
        FindTarget = True
      Case 2
'       --company
        RecordPosition = GetIniLong(CStr(target), "Companies", IdxFile)
        FindTarget = True
      Case 4
'       --holding company
        RecordPosition = GetIniLong(CStr(target), "Holding Companies", IdxFile)
        FindTarget = True
      Case Else
'       --cbg
        FindTarget = False
    End Select
    ShowStatusMessage "summarizing " & FileName & "   [" & LCase(target) & "]"
  Else
    ShowStatusMessage "summarizing " & FileName
  End If

' --set single business line factor
  State = FullStateName(FileName)
  RtnString = GetIniString(FactorIniFile, "Single Business Line Table", State)
  SingleBusLineFctr = CDbl(RtnString)

  stream = FreeFile()
  Open FileName For Input Shared As #stream

  estream = FreeFile()
  Open ErrorFile For Output As #estream

' --assign dstream even if detail is not saved, makes calls to ApplySegmentSupport e
asier
  dstream = FreeFile()

  If SaveDetail Then
'   --build root of file name
    DetailFile = Left$(FileName, InStr(FileName, ".")) & "cbg"
    Open DetailFile For Output As #dstream
  End If

' --move to correct file position
  If FindTarget Then
```

```
    If RecordPosition > 0 Then
      Seek #stream, RecordPosition
    Else
      Summarize = False
      Close stream, estream, dstream
      Exit Function
    End If
  End If

  Do Until EOF(stream)
    Line Input #stream, buffer
    If Len(buffer) < 150 Then
      NumberOfErrors = NumberOfErrors + 1
      Print #estream, buffer
    Else
      cbgRecord = ParseRecord(buffer, 48)
      If DoWholeFile Then
        --process every record, not looking for any matches
        ApplySegmentSupport cbgRecord, dstream
        CumulateCBGData cbgRecord
        CumulateHouseholds cbgRecord
        MinMaxLoopLengths cbgRecord
        CountHighCostLines cbgRecord
      Else
        --doing some kind of pattern search
        If FindTarget Then
          If Trim$(cbgRecord(FieldNumber)) = target Then
            --target is not CBG
            ApplySegmentSupport cbgRecord, dstream
            CumulateCBGData cbgRecord
            CumulateHouseholds cbgRecord
            MinMaxLoopLengths cbgRecord
            CountHighCostLines cbgRecord
          Else
            Exit Do
          End If
        Else
          --looking for specific cbg
          If Trim$(cbgRecord(6)) = target Then
            --set flag so you only read minimum number of records
            FoundCBG = True
            ApplySegmentSupport cbgRecord, dstream
            CumulateCBGData cbgRecord
            CumulateHouseholds cbgRecord
            MinMaxLoopLengths cbgRecord
            CountHighCostLines cbgRecord
          Else
            If FoundCBG Then Exit Do
          End If
        End If
      End If
    End If
  Loop

  Close stream, estream, dstream

  ShowStatusMessage

  -- notify that errors occured
  If NumberOfErrors Then
```

```
    MsgBox "Errors occured during processing, check " & ErrorFile & " for details",
, "BCPM Message"
  End If

End Function

Public Sub CountHighCostLines(Record As Variant)

  If Record(CBGLinesCol) <> 0 Then
    If Record(LoopCostCol) / Record(CBGLinesCol) > 10000 Then
      CumulativeLines = CumulativeLines + Record(CBGLinesCol)
    End If
  End If

End Sub

Public Sub CumulateHouseholds(Record As Variant)
Dim index           As Long

  index = GetByCostIndex
  HHByCost(index) = HHByCost(index) + Record(HHCol)

  index = GetByLoopIndex(Record(LoopCol))
  HHBYLoop(index) = HHBYLoop(index) + Record(HHCol)

End Sub

Public Sub MinMaxLoopLengths(Record As Variant)

  If Record(LoopCol) < minLoopLength Then
    minLoopLength = Record(LoopCol)
  End If

  If Record(LoopCol) > maxLoopLength Then
    maxLoopLength = Record(LoopCol)
  End If

End Sub

Public Sub RollupFile(FileName As String, Optional Targets As Variant, Optional Fiel
dNumber As Variant)
Dim index               As Long
Dim opt1                As OptionButton
Dim FoundSomething      As Boolean

  FoundSomething = False

  Initialize_Rollup

  Set opt1 = ThisWorkbook.Sheets("Options").OptionButtons("AggregateSupportCBG")

  If opt1.Value = xlOn Then
    If IsMissing(Targets) Then
        If Summarize(FileName) Then
          FoundSomething = True
        End If
    Else
      For index = 1 To UBound(Targets)
        If Summarize(FileName, Targets(index), FieldNumber) Then
          FoundSomething = True
```

```
        End If
      Next index
    End If
  Else
    If IsMissing(Targets) Then
      If SummarizeByWireCenter(FileName) Then
        FoundSomething = True
      End If
    Else
      For index = 1 To UBound(Targets)
        If SummarizeByWireCenter(FileName, Targets(index), FieldNumber) Then
          FoundSomething = True
        End If
      Next index
    End If
  End If

  If FoundSomething Then
    StatesUsed(1) = Left(Right(ExtractPath(FileName), 3), 2)
  End If

  FormatAggregateSupportSheet

  WriteData ThisWorkbook.Path & "\reports\rptData.csv"

  Set opt1 = Nothing

End Sub

Public Sub RollupFiles(Filenames() As String, Optional target As Variant, Optional F
ieldNumber As Variant)
Dim index          As Long
Dim opt1           As OptionButton


  Initialize_Rollup

  Set opt1 = ThisWorkbook.Sheets("Options").OptionButtons("AggregateSupportCBG")

  If opt1.Value = xlOn Then
    If IsMissing(target) Then
      For index = 1 To UBound(Filenames)
        If Summarize(Filenames(index)) Then
          ReDim Preserve StatesUsed(StateIndex)
          StatesUsed(StateIndex) = Left(Right(ExtractPath(Filenames(index)), 3), 2)
          StateIndex = StateIndex + 1
        End If
      Next index
    Else
      For index = 1 To UBound(Filenames)
        If Summarize(Filenames(index), target, FieldNumber) Then
          ReDim Preserve StatesUsed(StateIndex)
          StatesUsed(StateIndex) = Left(Right(ExtractPath(Filenames(index)), 3), 2)
          StateIndex = StateIndex + 1
        End If
      Next index
    End If
  Else
    If IsMissing(target) Then
      For index = 1 To UBound(Filenames)
```