```
        If SummarizeByWireCenter(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 SummarizeByWireCenter(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
  End If

  FormatAggregateSupportSheet

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

  Set opt1 = Nothing

End Sub

Public Sub AddToWireCenterRecord(myRecord As Variant)
Dim index              As Long

' --copy text fields
  For index = 1 To NumberOfStringFields
    wcRecord(index) = myRecord(index)
  Next index

' --cumulate all other fields
  For index = NumberOfStringFields + 1 To NumberOfInputFields
    wcRecord(index) = wcRecord(index) + CDbl(myRecord(index))
  Next index

End Sub

Public Sub CumulateCBGData(Record As Variant)
Dim col                As Long, _
    index              As Long, _
    costIndex          As Long, _
    Offset             As Long, _
    someNumber         As Long, _
    costShift          As Long

  col = GetIniLong(CStr(Record(DensityCol)), "Density", ModelIniFile)

  Offset = NumberOfStringFields

' --first cumulate add data in record
  For index = NumberOfStringFields + 1 To NumberOfInputFields
    Totals(CSIndex, col, index - Offset) = Totals(CSIndex, col, index - Offset) + CD
bl(Record(index))
  Next index

  Offset = NumberOfInputFields - NumberOfStringFields
  someNumber = UBound(BenchMarkSupport)
```

```
' --next post benchmark support amounts
  For costIndex = 1 To 2
    costShift = (costIndex - 1) * someNumber
    For index = 1 To someNumber
      Totals(CSIndex, col, index + Offset + costShift) = Totals(CSIndex, col, index
+ Offset + costShift) + BenchMarkSupport(index, costIndex)
    Next index
  Next costIndex

End Sub

Public Function GetByCostIndex() As Long
Dim PrelimIndex     As Integer
Dim NewAmount     As Long

' --convert monthly cost to cents
  NewAmount = CLng(CBGMonthlyCost * 100)

    Select Case NewAmount
      Case 0 To 499
        PrelimIndex = 1
      Case 500 To 999
        PrelimIndex = 2
      Case 1000 To 1499
        PrelimIndex = 3
      Case 1500 To 1999
        PrelimIndex = 4
      Case 2000 To 2499
        PrelimIndex = 5
      Case 2500 To 2999
        PrelimIndex = 6
      Case 3000 To 3499
        PrelimIndex = 7
      Case 3500 To 3999
        PrelimIndex = 8
      Case 4000 To 4499
        PrelimIndex = 9
      Case 4500 To 4999
        PrelimIndex = 10
      Case 5000 To 5499
        PrelimIndex = 11
      Case 5500 To 5999
        PrelimIndex = 12
      Case 6000 To 6499
        PrelimIndex = 13
      Case 6500 To 6999
        PrelimIndex = 14
      Case 7000 To 7499
        PrelimIndex = 15
      Case 7500 To 9999
        PrelimIndex = 16
      Case 10000 To 14999
        PrelimIndex = 17
      Case 15000 To 19999
        PrelimIndex = 18
      Case 20000 To 24999
        PrelimIndex = 19
      Case 25000 To 29999
        PrelimIndex = 20
```

```
      Case 30000 To 49999
        PrelimIndex = 21
      Case 50000 To 999999
        PrelimIndex = 22
      Case Else
        PrelimIndex = 23
    End Select

  GetByCostIndex = PrelimIndex
  'Debug.Print CBGMonthlyCost, CInt(NewAmount / 5), PrelimIndex
End Function

Private Function GetByLoopIndex(LoopLength As Long) As Long
Dim PrelimIndex     As Integer
Dim NewAmount       As Long
' --use integer division to take first cut at index

  NewAmount = CLng(LoopLength)

    Select Case NewAmount
      Case 0 To 4999
        PrelimIndex = 1
      Case 5000 To 9999
        PrelimIndex = 2
      Case 10000 To 14999
        PrelimIndex = 3
      Case 15000 To 19999
        PrelimIndex = 4
      Case 20000 To 24999
        PrelimIndex = 5
      Case 25000 To 29999
        PrelimIndex = 6
      Case 30000 To 39999
        PrelimIndex = 7
      Case 40000 To 49999
        PrelimIndex = 8
      Case 50000 To 59999
        PrelimIndex = 9
      Case 60000 To 69999
        PrelimIndex = 10
      Case 70000 To 79999
        PrelimIndex = 11
      Case 80000 To 89999
        PrelimIndex = 12
      Case 90000 To 99999
        PrelimIndex = 13
      Case 100000 To 149999
        PrelimIndex = 14
      Case 150000 To 199999
        PrelimIndex = 15
      Case Else
        PrelimIndex = 16
    End Select

  GetByLoopIndex = PrelimIndex

End Function

Public Sub WriteData(OutputFile As String)
Dim stream          As Long, _
```

```
    row              As Long, _
    col              As Long, _
    Item             As Long, _
    MaxDataItems     As Long

  stream = FreeFile()
  Open OutputFile For Output As #stream

  MaxDataItems = (NumberOfInputFields - NumberOfStringFields) + 2 * Benchmarks + 2

' --write first block of data
  For row = 1 To NumberOfCompanySizes
    Write #stream, "Company Size: " & CompanySizeLabels(row)
    For col = 1 To NumberOfDensityGroups
      Write #stream, DensityGroupLabels(col);
      For Item = 1 To MaxDataItems
        Write #stream, Totals(row, col, Item);
      Next Item
      Write #stream,
    Next col
  Next row

  Print #stream,

  For row = 1 To UBound(HHByCost)
    Print #stream, CStr(HHByCost(row))
  Next row

  Print #stream,

  For row = 1 To UBound(HHBYLoop)
    Print #stream, CStr(HHBYLoop(row))
  Next row

  Print #stream,
  Print #stream, minLoopLength
  Print #stream, maxLoopLength
  Print #stream, CumulativeLines
  Print #stream, SingleBusinessLines
  Close stream

End Sub

Public Sub PopulateBenchmarkDollars()
Dim rtn          As Long, _
    index        As Long

  rtn = -1
  index = 0

  While rtn <> 0
    index = index + 1
'   --GetPrivateProfileInt returns -1 when not successful
    rtn = GetPrivateProfileInt("Aggregate Support", CStr(index), 0, FactorIniFile)
    If rtn > 0 Then
      ReDim Preserve BenchmarkDollars(index)
      BenchmarkDollars(index) = rtn
    End If
  Wend
```

```
' --set private data member
  Benchmarks = index - 1

End Sub


Private Sub PopulateCapCostFactors()
Dim index        As Long
Dim AccntDesc    As String, _
    RtnString    As String
Dim Record       As Variant

  For index = 1 To NumberOfAccounts
'   --get account description
    AccntDesc = GetIniString(ModelIniFile, "Accounts", CStr(index))
'   --get capital cost factor
    RtnString = GetIniString(FactorIniFile, "Account Capital Cost", AccntDesc)
    Record = ParseRecord(RtnString, 7)
    CapCostFactors(index) = CDbl(Record(7))
  Next index

End Sub

Public Sub Initialize_Rollup()
Dim dd              As DropDown
Dim rng             As Range

  NumberOfAccounts = 19

  NumberOfInputFields = 48
  NumberOfStringFields = 6

  minLoopLength = 1000000
  maxLoopLength = 0
  CumulativeLines = 0
  SingleBusinessLines = 0

  ValidationRow = 4

  ReDim Record(NumberOfInputFields)
  ReDim SegmentDollars(NumberOfAccounts)
  ReDim InputSource(NumberOfAccounts, 2)
  ReDim SupportFactors(NumberOfAccounts, 3)
  ReDim CapCostFactors(NumberOfAccounts)
  ReDim HHByCost(23)
  ReDim HHBYLoop(16)

  ReDim StatesUsed(1)
  StateIndex = 1

  ReDim CompanySizeLabels(NumberOfCompanySizes)
  CompanySizeLabels(1) = "Small"
  CompanySizeLabels(2) = "Medium"
  CompanySizeLabels(3) = "Large"

  ReDim DensityGroupLabels(NumberOfDensityGroups)
  DensityGroupLabels(1) = "0 to 10"
  DensityGroupLabels(2) = "11 to 50"
  DensityGroupLabels(3) = "51 to 150"
  DensityGroupLabels(4) = "151 to 500"
```

```
  DensityGroupLabels(5) = "501 to 2000"
  DensityGroupLabels(6) = "2001 to 5000"
  DensityGroupLabels(7) = "Greater 5001"

  ErrorFile = "c:\errors.bcp"
  ModelIniFile = ThisWorkbook.Sheets("RangeNames").Range("Model_Inf_File")

  Set dd = ThisWorkbook.Sheets("Reports").DropDowns("FactorSets")
  FactorIniFile = ThisWorkbook.Sheets("RangeNames").Range("Factor_Path") & dd.List(d
d.ListIndex)
  Set dd = Nothing

  PopulateBenchmarkDollars
  PopulateCapCostFactors
  PopulateInputSource
  PopulateExpenses
  PopulateSupportFactors

  SetColumns

  ShowDetail = (ThisWorkbook.Sheets("RangeNames").Range("ShowAggregateSupport").Valu
e = True)

  Set rng = ThisWorkbook.Sheets("Aggregate Support Calculations").Range("ascDataArea
")
  rng.ClearContents

  ReDim Totals(NumberOfCompanySizes, NumberOfDensityGroups, NumberOfInputFields - Nu
mberOfStringFields + 2 * Benchmarks + 2)

  Set rng = Nothing

End Sub

Private Sub PopulateInputSource()
Dim index           As Long
Dim RtnString       As String
Dim columns         As Variant

  For index = 1 To NumberOfAccounts
    RtnString = GetIniString(ModelIniFile, "Accounts", CStr(index))
    --next find the corresponding column in Data
    RtnString = GetIniString(ModelIniFile, "Column In Output", RtnString)
    columns = ParseRecord(RtnString, 2)
    InputSource(index, 1) = CLng(columns(1))
    InputSource(index, 2) = CLng(columns(2))
  Next index

End Sub

Private Sub PopulateExpenses()
Dim RtnString              As String
Dim MonthlyExpenses        As Variant

  RtnString = GetIniString(FactorIniFile, "Monthly Operating Expenses - Residential"
, "Total Expense")
  If RtnString <> "Not Found" Then
    MonthlyExpenses = ParseRecord(RtnString, 3)
    Expenses(1) = CDbl(MonthlyExpenses(1))
    Expenses(2) = CDbl(MonthlyExpenses(2))
```

```
    Expenses(3) = CDbl(MonthlyExpenses(3))
  Else
    MsgBox "Problem loading Monthly Operating Expenses, check " & FactorIniFile, , "
BCPM Message"
    EndApplication
  End If

End Sub

Private Sub SetColumns()

  LoopCol = GetIniLong("LoopLength", "Output Columns", ModelIniFile)
  HHCol = GetIniLong("Households", "Output Columns", ModelIniFile)
  CompanySizeCol = GetIniLong("CoSize", "Output Columns", ModelIniFile)
  CBGLinesCol = GetIniLong("CBGLines", "Output Columns", ModelIniFile)
  LoopCostCol = GetIniLong("LoopCost", "Output Columns", ModelIniFile)
  DensityCol = GetIniLong("Density", "Output Columns", ModelIniFile)
  BusLinesCol = GetIniLong("BusLines", "Output Columns", ModelIniFile)

End Sub

Private Sub PopulateSupportFactors()
Dim index          As Long, _
    CSIndex        As Long
Dim AccntDesc      As String, _
    RtnString      As String
Dim Factors        As Variant

  For index = 1 To NumberOfAccounts
    --get account description
    AccntDesc = GetIniString(ModelIniFile, "Accounts", CStr(index))

    --check to see if support factors exist
    RtnString = GetIniString(FactorIniFile, "Support Ratio Table", AccntDesc)
    If RtnString <> "Not Found" Then
      Factors = ParseRecord(RtnString, 3)
      SupportFactors(index, 1) = CDbl(Factors(1))
      SupportFactors(index, 2) = CDbl(Factors(2))
      SupportFactors(index, 3) = CDbl(Factors(3))
    Else
      SupportFactors(index, 1) = 1
      SupportFactors(index, 2) = 1
      SupportFactors(index, 3) = 1
    End If
  Next index

End Sub
```

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

Private LastWorkbook                As Workbook
Private WorkbookPath                As String

Public Sub AddRangeNamesToList()
Dim NameList            As Listbox
Dim SheetList           As Listbox
Dim RangeName           As Object

  Set NameList = ThisWorkbook.Sheets("Input Tables").ListBoxes("lstRangeNames")
  Set SheetList = ThisWorkbook.Sheets("Input Tables").ListBoxes("lstWorksheets")

  NameList.RemoveAllItems

  For Each RangeName In LastWorkbook.Names
    If InStr(RangeName, LastWorkbook.Sheets(SheetList.List(SheetList.ListIndex)).Nam
e) Then
      If InStr(RangeName.Name, "print") = 0 Then
        NameList.AddItem RangeName.Name
      End If
    End If
  Next RangeName

End Sub

Public Sub AddActiveWorkbookToList()
Dim sht                 As Worksheet
Dim lst                 As Listbox
Dim index               As Long
Dim FoundWB             As Boolean
Dim target              As String
Dim newWb               As Workbook


  For Each newWb In Application.Workbooks
    If newWb.Name <> ThisWorkbook.Name Then
      Set LastWorkbook = newWb
      Exit For
    End If
  Next newWb

  target = LastWorkbook.Name

  Set sht = ThisWorkbook.Sheets("Input Tables")
  Set lst = sht.ListBoxes("lstWorkbooks")

  For index = 1 To lst.ListCount
    If lst.List(index) = target Then
      FoundWB = True
      Exit For
    End If
  Next index

  If FoundWB Then
    lst.ListIndex = index
  Else
    lst.AddItem target
    lst.ListIndex = lst.ListCount
  End If
```

```
End Sub

Public Sub ClearWorkbookSelection()
Dim lst                 As Listbox
Dim sht                 As Worksheet

  Set sht = ThisWorkbook.Sheets("Input Tables")
  Set lst = sht.ListBoxes("lstWorkbooks")

  If lst.ListCount > 0 Then
'   --clear selections in Workbooks list
    lst.ListIndex = 0
'   --clear other list if necessary
    sht.ListBoxes("lstWorksheets").RemoveAllItems
    sht.ListBoxes("lstRangeNames").RemoveAllItems
  End If

  Set sht = Nothing
  Set lst = Nothing

End Sub

Public Sub ClearWorksheetSelection()
Dim lst                 As Listbox
Dim sht                 As Worksheet

  Set sht = ThisWorkbook.Sheets("Input Tables")
  Set lst = sht.ListBoxes("lstWorksheets")

  If lst.ListCount > 0 Then
'   --clear selections in Worksheets list
    lst.ListIndex = 0
'   --clear other list if necessary
    sht.ListBoxes("lstRangeNames").RemoveAllItems
  End If

  Set sht = Nothing
  Set lst = Nothing

End Sub

Public Sub ClearRangeNameSelection()

  ThisWorkbook.Sheets("Input Tables").ListBoxes("lstRangeNames").ListIndex = 0

End Sub

Public Sub InputTables_SetPrimaryDisplay()
Dim btn                     As OptionButton
Dim sht                     As Worksheet
Dim lst                     As Listbox

  Set sht = ThisWorkbook.Sheets("Input Tables")
  Set btn = sht.OptionButtons("optInvestmentInputs")
  Set lst = sht.ListBoxes("lstWorkbooks")

  sht.ListBoxes("lstWorksheets").RemoveAllItems
  sht.ListBoxes("lstRangeNames").RemoveAllItems

  If btn.Value = xlOn Then
'   --build display for investment inputs
```

```
    WorkbookPath = ThisWorkbook.Path & "\logic\"
    GetFileNames (WorkbookPath & "*.xls")
    lst.List = p_FileNames
  Else
    --build display for capcost option
    WorkbookPath = ThisWorkbook.Path & "\factors\"
    GetFileNames (WorkbookPath & "*.xls")
    lst.List = p_FileNames
  End If

  Set LastWorkbook = Nothing

  Set sht = Nothing
  Set btn = Nothing
  Set lst = Nothing

End Sub

Public Sub InputTables_CloseWorkbook()
Dim btn                 As OptionButton
Dim sht                 As Worksheet
Dim rtn                 As Variant

  If LastWorkbook Is Nothing Then
    MsgBox "Must Load workbook first"
    Exit Sub
  End If

  Set sht = ThisWorkbook.Sheets("Input Tables")
  Set btn = sht.OptionButtons("optInvestmentInputs")

  Application.ScreenUpdating = False

  If btn.Value = xlOn Then
    --close investment inputs workbook
    SaveWorkbook LastWorkbook
    With LastWorkbook
      .Saved = True
      .Close
    End With
  Else
    --invoke close routine in "Capcost" workbook
    On Local Error Resume Next
    If LastWorkbook.Saved = False Then
      rtn = Run(LastWorkbook.Name & "!mnuSaveView")
    End If
    --workbook saved via CAPCOST menu choice
    If Err = 424 Then
      Err = 0
      Set LastWorkbook = Nothing
      Exit Sub
    End If
    On Local Error GoTo 0
    rtn = Run(LastWorkbook.Name & "!mnuCloseModel")
  End If

  Set LastWorkbook = Nothing
  ClearWorkbookSelection

  Set sht = Nothing
  Set btn = Nothing
```

```
End Sub

Public Sub InputTables_SaveWorkbook()
Dim btn                 As OptionButton
Dim sht                 As Worksheet
Dim rtn                 As Variant
Dim lst                 As Listbox
Dim ModelName           As String

  If LastWorkbook Is Nothing Then
    MsgBox "Must Load workbook first"
    Exit Sub
  End If

  Set sht = ThisWorkbook.Sheets("Input Tables")
  Set btn = sht.OptionButtons("optInvestmentInputs")
  Set lst = sht.ListBoxes("lstWorkbooks")

  If btn.Value = xlOn Then
'   --saving investment input workbook
    ModelName = Application.InputBox(prompt:="Enter an 8 character name", _
                                     Title:="BCPM Logic Set", _
                                     Type:=2)
    If ModelName = "False" Then Exit Sub
    ModelName = Trim(ModelName)
    If Len(ModelName) > 8 Then
      ModelName = Left$(ModelName, 8)
    End If
    If (ModelName = "BCPM") Or (ModelName = "BCPM.xls") Then
      MsgBox ModelName & " is reserved, choose another name"
      Exit Sub
    End If
    SaveWorkbook LastWorkbook, ModelName
  Else
'   --invoke save routine in "Capcost" workbook
    rtn = Run(LastWorkbook.Name & "!mnuSaveView")
  End If

  AddActiveWorkbookToList

  Set sht = Nothing
  Set btn = Nothing

End Sub

Public Sub InputTables_ShowWorkbook()
Dim sht                 As Worksheet
Dim lst1                As Listbox, _
    lst2                As Listbox

  Set sht = ThisWorkbook.Sheets("Input Tables")
  Set lst1 = sht.ListBoxes("lstWorksheets")
  Set lst2 = sht.ListBoxes("lstRangeNames")

  If LastWorkbook Is Nothing Then
    MsgBox "Workbook must be loaded first"
    Exit Sub
  End If

  p_ReturningFromWB = True
```

```
  If lst2.ListIndex > 0 Then
'   --goto selected name
    Application.Goto reference:=LastWorkbook.Sheets(lst1.List(lst1.ListIndex)).Range
(lst2.List(lst2.ListIndex)), scroll:=True
  Else
    If lst1.ListIndex > 0 Then
'     --goto selected sheet
      Application.Goto reference:=LastWorkbook.Sheets(lst1.List(lst1.ListIndex)).Cel
ls(1, 1), scroll:=True
    Else
'     --goto first sheet by default
      Application.Goto reference:=LastWorkbook.Worksheets(1).Cells(1, 1), scroll:=Tr
ue
    End If
  End If

  Set sht = Nothing
  Set lst1 = Nothing
  Set lst2 = Nothing

End Sub

Public Sub InputTables_LoadWorkbook()
Dim sht                 As Worksheet
Dim lst                 As Listbox
Dim shts()              As String

  Set sht = ThisWorkbook.Sheets("Input Tables")
  Set lst = sht.ListBoxes("lstWorkbooks")

  Application.ScreenUpdating = False

  If lst.ListIndex < 1 Then
    MsgBox "Must select workbook"
    Exit Sub
  Else
'   --close all open workbooks
    CloseAllOpenWorkbooks
'   --open the selected workbook, populate worksheets list
    Workbooks.Open FileName:=WorkbookPath & lst.List(lst.ListIndex)
    ActiveWorkbook.RunAutoMacros xlAutoOpen
    Set LastWorkbook = ActiveWorkbook
    sht.Activate
    InputTables_LoadWorksheetNames
  End If

  Set sht = Nothing
  Set lst = Nothing

End Sub

Public Sub InputTables_LoadWorksheetNames()
Dim ws                  As Worksheet
Dim shts()              As String
Dim index               As Long

  For Each ws In LastWorkbook.Worksheets
    If InStr(ws.Name, "Inputs") Then
      index = index + 1
      ReDim Preserve shts(index)
```

```
      shts(index) = ws.Name
    End If
  Next ws

  ThisWorkbook.Sheets("Input Tables").ListBoxes("lstWorksheets").List = shts

End Sub
```