```
  DeleteSection "Holding Companies", IndexFile
  DeleteSection "Companies", IndexFile

' --now process input file, writing entry every time there is a change in wire cente
r
  Do Until EOF(InputStream)
    RecordLocation = Seek(InputStream)
    Line Input #InputStream, buffer
'   --only process nonblank records
    If Len(buffer) > position Then
      If PassColumns Then
        Record = ParseRecord(buffer, columns)
      Else
        Record = ParseRecord(buffer)
      End If
      If Record(WireCenterCol) <> HoldWireCenter Then
        WriteIniString IndexFile, "Wire Centers", Record(WireCenterCol), CStr(Record
Location)
        HoldWireCenter = Record(WireCenterCol)
      End If
      If Record(ParentCol) <> HoldParent Then
        WriteIniString IndexFile, "Holding Companies", Record(ParentCol), CStr(Recor
dLocation)
        HoldParent = Record(ParentCol)
      End If
      If Record(CompanyCol) <> HoldCompany Then
        WriteIniString IndexFile, "Companies", Record(CompanyCol), CStr(RecordLocati
on)
        HoldCompany = Record(CompanyCol)
      End If
    End If
  Loop

' --close all files
  Close InputStream

  FlushCache IndexFile

  BuildFileIndexes = IndexFile

  ShowStatusMessage

End Function

'==================================================================================
============
Public Function ExtractName(FileSpec As String, Optional BaseOnly As Variant) As Str
ing
Dim IncludeExtension          As Boolean
Dim index                     As Long
Dim SpecOut                   As String

  If IsMissing(BaseOnly) Then
    IncludeExtension = True
  Else
    IncludeExtension = BaseOnly
  End If

  On Local Error Resume Next

  If InStr(FileSpec, "\") Then
```

```
    For index = Len(FileSpec) To 1 Step -1
      If Mid$(FileSpec, index, 1) = "\" Then
        SpecOut = Mid$(FileSpec, index + 1)
        Exit For
      End If
    Next index
  Else
    If InStr(FileSpec, ":") = 2 Then
      SpecOut = Mid$(FileSpec, 3)
    Else
      SpecOut = FileSpec
    End If
  End If

  If Not IncludeExtension Then
    index = InStr(SpecOut, ".")
    If index Then
      SpecOut = Left(SpecOut, index - 1)
    End If
  End If

  ExtractName = SpecOut

End Function

'==========================================================================
============
Public Function ExtractPath(FileSpec As String) As String
Dim index             As Long
Dim SpecOut           As String

  On Local Error Resume Next

  If InStr(FileSpec, "\") Then
    For index = Len(FileSpec) To 1 Step -1
      If Mid$(FileSpec, index, 1) = "\" Then
        SpecOut = Left(FileSpec, index)
        Exit For
      End If
    Next index
  ElseIf InStr(FileSpec, ":") = 2 Then
      SpecOut = CurDir$(FileSpec)
      If Len(SpecOut) = 0 Then SpecOut = CurDir()
  Else
    SpecOut = CurDir()
  End If

  If Right$(SpecOut, 1) <> "\" Then
    SpecOut = SpecOut & "\"
  End If

  ExtractPath = SpecOut

End Function
```

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

'============================DATA DECLARATIONS=====================================
===

Public Type ModelRecord
  Name As String
  LogicSet As String
  CBGDataSet As String
  Description As String
End Type

Public CurrentModel              As ModelRecord

'============================SUBROUTINES/FUNCTIONS=================================
=======

Public Sub SetupNewModel()
Dim ModelName          As String
Dim ds                 As DialogSheet
Dim dd                 As DropDown

  ModelName = Application.InputBox(prompt:="Enter an 8 character model name", _
                                   Title:="BCPM New Model", _
                                   Type:=2)

  If ModelName = "False" Then Exit Sub

  ModelName = Trim(ModelName)

  If Len(ModelName) Then
    Set ds = ThisWorkbook.Sheets("dlgNewModel")
    Set dd = ds.DropDowns("ModelName")
'   --truncate name if more than 8 characters
    If Len(ModelName) > 8 Then
      ModelName = Left$(ModelName, 8)
    End If
    ClearModelInformation
    dd.Caption = ModelName
    FormatNewModelDialog
    Set ds = Nothing
    Set dd = Nothing
 End If

End Sub

Public Function GetModelParameters(FileName As String, ModelName As String) As Varia
nt
Dim Information(3)      As Variant

' --get input data
  Information(1) = GetIniString(FileName, ModelName, "CBG Data Set")
  Information(2) = GetIniString(FileName, ModelName, "Logic Set")
  Information(3) = GetIniString(FileName, ModelName, "description")

  GetModelParameters = Information

End Function

Private Sub ClearModelInformation()
```

```
  CurrentModel.Name = ""
  CurrentModel.CBGDataSet = ""
  CurrentModel.LogicSet = ""
  CurrentModel.Description = ""

End Sub

Private Function CheckModelInfo() As Boolean
Dim index           As Long
Dim ds              As DialogSheet
Dim eb              As EditBox
Dim dd              As DropDown

  Set ds = ThisWorkbook.Sheets("dlgNewModel")
  CheckModelInfo = False

  Set eb = ds.EditBoxes("CBGDataSet")
  If Len(eb.Text) = 0 Then
    MsgBox "Must enter CBG Data Set name"
    Exit Function
  End If

  Set eb = ds.EditBoxes("description")
  If Len(eb.Text) = 0 Then
    MsgBox "Must enter a model description"
    Exit Function
  End If

  CheckModelInfo = True

End Function

Public Sub ClearNewModelDialog()
Dim ds              As DialogSheet
Dim cntl            As Object

  Set ds = ThisWorkbook.Sheets("dlgNewModel")

  On Local Error Resume Next

  For Each cntl In ds.DropDowns
    cntl.ListIndex = 0
    cntl.Text = ""
  Next cntl

  For Each cntl In ds.EditBoxes
    cntl.Text = ""
  Next cntl

End Sub

Public Sub DeleteModel()
Dim ModelFile         As String
Dim dd                As DropDown

  Set dd = ThisWorkbook.Sheets("dlgNewModel").DropDowns("ModelName")

  ModelFile = ThisWorkbook.Sheets("RangeNames").Range("Model_Inf_File")
' --update models.inf
  DeleteEntry "Models", dd.Caption, ModelFile
  DeleteSection dd.Caption, ModelFile
```

```
Public Sub SwitchToInputTablesTab()

  Application.DisplayFormulaBar = True
  Windows("Control").Activate
  InstructionsSheet_GotFocus
  Sheets("Input Tables").Activate

End Sub

Public Sub SwitchToInstructionTab()

  Windows("Control").Activate
  InstructionsSheet_GotFocus
  Sheets("Instructions").Activate

End Sub

Public Sub SwitchToReviewTab()

  Windows("Control").Activate
  ReviewSheet_GotFocus
  Sheets("Review").Activate

End Sub

Public Sub SwitchToOptionsTab()

  Windows("Control").Activate
  OptionsSheet_GotFocus
  Sheets("Options").Activate

End Sub

Public Sub SwitchToReportsTab()

  Windows("Control").Activate
  ReportsSheet_GotFocus
  Sheets("Reports").Activate

End Sub

Public Sub SwitchToProcessTab()

  Windows("Control").Activate
  ProcessSheet_GotFocus
  Sheets("Process").Activate

End Sub


Public Sub mnuCapCost()
Dim dlg                 As DialogSheet
Dim lst                 As Listbox
Dim FileSpec            As String, _
    FactorPath          As String

  Set dlg = ThisWorkbook.DialogSheets("dlgCapCostWorkbooks")
  Set lst = dlg.ListBoxes("lstWorkbooks")

  lst.RemoveAllItems
```

```
  Set dlg = Nothing
  Set lstSheets = Nothing
  Set FromSheet = Nothing

End Sub

Public Sub mnuExit()

  CloseAllOpenWorkbooks
  CloseControlWorkbook

End Sub

'=================================================================================
============
Public Sub RemoveMenus()
Dim index           As Integer
Dim MenuNames(3)    As Integer

    MenuNames(1) = xlWorksheet
    MenuNames(2) = xlModule
    MenuNames(3) = xlNoDocuments

    On Error Resume Next
    For index = 1 To 3
        Application.MenuBars(MenuNames(index)).Menus(MENU_TITLE).Delete
    Next index

End Sub
```

```
  ClearNewModelDialog
  UpdateModelList

  Set dd = Nothing

End Sub

Public Sub FormatNewModelDialog()
Dim index           As Long
Dim FileSpec        As String
Dim ds              As DialogSheet
Dim dd              As DropDown

  Set ds = ThisWorkbook.Sheets("dlgNewModel")

' --make sure model info is in sync with Process sheet
  Set dd = ds.DropDowns("ModelName")

  SetCurrentModel dd.Caption

' --set up Logic Sets dropdown
  Set dd = ds.DropDowns("LogicSets")
  dd.RemoveAllItems
  FileSpec = ThisWorkbook.Sheets("RangeNames").Range("Model_Path") & "*.xls"
  If Not GetFileNames(FileSpec) Then
    ShowMessage "Message: No workbooks in " & ThisWorkbook.Sheets("RangeNames").Rang
e("Model_Path") _
                & Chr$(10) & "Action: Unable to create new model"
    Exit Sub
  End If
  For index = 1 To UBound(p_FileNames)
    dd.AddItem p_FileNames(index)
  Next index

' --select current model settings by default
  If CurrentModel.LogicSet = "Not Found" Then
    SelectListItem dd, "bcpm.xls"
  Else
    SelectListItem dd, CurrentModel.LogicSet
  End If

' --setup editboxes
  If CurrentModel.Description = "Not Found" Then
    ds.EditBoxes("Description").Text = ""
  Else
    ds.EditBoxes("Description").Text = CurrentModel.Description
  End If

  If CurrentModel.CBGDataSet = "Not Found" Then
    ds.EditBoxes("CBGDataSet").Text = "base.csv"
  Else
    ds.EditBoxes("CBGDataSet").Text = CurrentModel.CBGDataSet
  End If

' --clear objects
  Set dd = Nothing
  Set ds = Nothing

End Sub
```

```
' --populate CurrentModel
  CurrentModel.LogicSet = ModelInfo(2)
  CurrentModel.CBGDataSet = ModelInfo(1)
  CurrentModel.Description = ModelInfo(3)

' --post model workbook name to range name
  ThisWorkbook.Sheets("RangeNames").Range("Model_WB_Name") = ThisWorkbook.Sheets("Ra
ngeNames").Range("Model_Path") & CurrentModel.LogicSet

End Sub

Public Sub ShowNewModelDialog()

  FormatNewModelDialog
  ThisWorkbook.Sheets("dlgNewModel").Show

End Sub

Public Sub SaveModelInfo()
Dim Section          As String, _
    FileName         As String, _
    Value            As String
Dim ds               As DialogSheet
Dim dd               As DropDown

  If Not CheckModelInfo Then
    Exit Sub
  End If

  Set ds = ThisWorkbook.Sheets("dlgNewModel")
  FileName = ThisWorkbook.Sheets("RangeNames").Range("Model_Inf_File")

' --Section should be the model name
  Section = ds.DropDowns("ModelName").Text

' --write entry to [Models] section
  WriteIniString FileName, "Models", Section, ""

' --now write entries to section
  Value = ds.EditBoxes("CBGDataSet").Text
  WriteIniString FileName, Section, "CBG Data Set", Value

' --write workbook
  Set dd = ds.DropDowns("LogicSets")
  Value = dd.List(dd.ListIndex)
  WriteIniString FileName, Section, "Logic Set", Value

' --write description
  Value = ds.EditBoxes("description").Text
  WriteIniString FileName, Section, "Description", Value

' --make saved model the current model
  UpdateModelList

  Set dd = Nothing
  Set ds = Nothing

End Sub

Public Sub UpdateModelList()
```

```
Dim Models          As Variant
Dim RangeAddress    As String, _
    FileName        As String
Dim rtn             As Long, _
    index           As Long
Dim rng             As Range, _
    NewRng          As Range

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

  rng.ClearContents
  FileName = ThisWorkbook.Sheets("RangeNames").Range("Model_Inf_File")

  rtn = SectionEntriesToArray(Models, "Models", FileName)
  Set NewRng = rng.Resize(rtn, 1)

  For index = 1 To UBound(Models)
    NewRng.Offset(index - 1, 0).Cells(1, 1).Value = Models(index)
  Next index

  RangeAddress = "=RangeNames!" & NewRng.Address

  ThisWorkbook.Names.Add Name:="Models", RefersTo:=RangeAddress

  Set rng = Nothing
  Set NewRng = Nothing

End Sub
```

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

Public Declare Function GetPrivateProfileInt& Lib "kernel32" Alias "GetPrivateProfil
eIntA" (ByVal AppName As String, ByVal KeyName As Any, ByVal Default As Long, ByVal
FileName As String)
Public Declare Function GetPrivateProfileString& Lib "kernel32" Alias "GetPrivatePro
fileStringA" (ByVal Section As String, ByVal KeyName As Any, ByVal Default As String
, ByVal buffer As String, ByVal buffersize As Long, ByVal FileName As String)
Public Declare Function WritePrivateProfileString& Lib "kernel32" Alias "WritePrivat
eProfileStringA" (ByVal Section As Any, ByVal KeyName As Any, ByVal Value As Any, By
Val FileName As String)

Private Const SECTION_BUFFER_SIZE = 8192
Public Sub FlushCache(IniFile As String)
Dim rtn         As Long

  rtn = WritePrivateProfileString(0&, 0&, 0&, IniFile)

End Sub

Public Sub DeleteEntry(Section As String, Entry As String, IniFile As String)
Dim rtn         As Long

  rtn = WritePrivateProfileString(Section, Entry, 0&, IniFile)

End Sub

Public Sub DeleteSection(Section As String, IniFile As String)
Dim rtn       As Long

  rtn = WritePrivateProfileString(Section, 0&, 0&, IniFile)

End Sub

Public Function GetIniLong(target As String, Optional SectionName As Variant, Option
al FileName As Variant) As Long
Static IniFile          As String
Static Section          As String
Dim rtn                 As Long

  If Not IsMissing(SectionName) Then
    Section = SectionName
  End If

  If Not IsMissing(FileName) Then
    IniFile = FileName
  End If

  GetIniLong = GetPrivateProfileInt(Section, target, -1, IniFile)

End Function

Public Function GetAllSettings(SectionName As String, IniFile As String) As Variant
Dim Settings()      As String
Dim stream          As Long, _
    position        As Long, _
    index           As Long
Dim buffer          As String
Dim FoundSection    As Boolean
```

```
  stream = FreeFile()
  Open IniFile For Input Shared As #stream

  Do Until EOF(stream)
    Line Input #stream, buffer
    If Len(buffer) Then
      If Not FoundSection Then
        FoundSection = (("[" & SectionName & "]") = buffer)
      Else
        If Left$(buffer, 1) = "[" Then Exit Do
        position = InStr(buffer, "=")
        index = index + 1
        ReDim Preserve Settings(index)
        Settings(index) = Left$(buffer, position - 1)
      End If
    End If
  Loop

  Close stream

  GetAllSettings = Settings

End Function

Public Function GetSectionEntries(SectionName As String, IniFile As String) As Strin
g
Dim buffer        As String
Dim rtn           As Long, _
    buffersize    As Long

  buffersize = SECTION_BUFFER_SIZE
  buffer = Space$(buffersize)

  rtn = GetPrivateProfileString(SectionName, 0&, "", buffer, buffersize, IniFile)
  GetSectionEntries = Trim$(Left$(buffer, rtn + 1))

End Function

Public Function SectionEntriesToArray(SectionEntries As Variant, SectionName As Stri
ng, IniFile As String) As Long
Dim buffer            As String
Dim Table()           As String
Dim Entries           As Long, _
    NullPosition      As Long

  buffer = GetSectionEntries(SectionName, IniFile)

  Entries = 0

  Do While Asc(buffer)
    NullPosition = InStr(buffer, Chr$(0))
    If NullPosition > 1 Then
      Entries = Entries + 1
      ReDim Preserve Table(Entries)
      Table(Entries) = Left$(buffer, NullPosition - 1)
    End If
    buffer = Mid$(buffer, NullPosition + 1)
  Loop

  SectionEntries = Table
```

```
  SectionEntriesToArray = Entries

End Function

Public Function GetSections(FileName As String) As Variant
Dim stream          As Long, _
    index           As Long
Dim buffer          As Variant, _
    Sections()      As Variant

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

  Do Until EOF(stream)
    Line Input #stream, buffer
    --section headers should start with a left bracket
    If Left(buffer, 1) = "[" Then
      index = index + 1
      ReDim Preserve Sections(index)
      Sections(index) = Mid(buffer, 2, Len(buffer) - 2)
    End If
  Loop

  Close stream

  GetSections = Sections

End Function

Public Function GetIniString(FileName As String, Section As String, Key As String) A
s Variant
Dim buffer          As String
Dim buffersize      As Long, _
    rtn             As Long

  buffersize = 2048
  buffer = Space(buffersize)

  rtn = GetPrivateProfileString(Section, Key, "Not Found", buffer, buffersize - 1, F
ileName)

  If rtn > 0 Then
    buffer = Left(buffer, rtn)
  Else
    buffer = "Not Found"
  End If

  GetIniString = buffer

End Function

Public Sub WriteIniString(FileName As String, Section As String, Key As String, Valu
e As String)
Dim rtn         As Long

  rtn = WritePrivateProfileString(Section, Key, Value, FileName)

End Sub
```

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

Public Sub ShowOptionsTab()

  p_ReturningFromOptions = True
  SwitchToOptionsTab

End Sub

Public Sub ResetDefaultOptions()
Dim chk1                As CheckBox, _
    chk2                As CheckBox, _
    chk3                As CheckBox, _
    chk4                As CheckBox
Dim opt1                As OptionButton, _
    opt2                As OptionButton
Dim lst                 As Listbox
Dim Choices()           As Boolean
Dim index               As Long

  Set lst = ThisWorkbook.Sheets("Options").ListBoxes("lstReports")
  Set chk1 = ThisWorkbook.Sheets("Options").CheckBoxes("SummaryReports")
  Set chk2 = ThisWorkbook.Sheets("Options").CheckBoxes("IndividualReports")
  Set chk3 = ThisWorkbook.Sheets("Options").CheckBoxes("AggregateSupportDetail")
  Set chk4 = ThisWorkbook.Sheets("Options").CheckBoxes("SaveReports")
  Set opt1 = ThisWorkbook.Sheets("Options").OptionButtons("AggregateSupportCBG")
  Set opt2 = ThisWorkbook.Sheets("Options").OptionButtons("AggregateSupportWC")

' --all reports are selected
  ReDim Choices(lst.ListCount)
  For index = 1 To lst.ListCount
    Choices(index) = True
  Next index
  lst.Selected = Choices

' --Aggregate Support is calculated by Census Block Group
  opt1.Value = True
  opt2.Value = False

' --only summary reports are printed
  chk1.Value = True
  chk2.Value = False
  chk3.Value = False
  chk4.Value = False

  Set lst = Nothing
  Set chk1 = Nothing
  Set chk2 = Nothing
  Set chk3 = Nothing
  Set chk4 = Nothing
  Set opt1 = Nothing
  Set opt2 = Nothing

End Sub
```

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

Public Sub SetCLLIQuadRange(formulaRange As Range)
Dim QuadRange          As Range

' --set reference to existing range
  Set QuadRange = ActiveWorkbook.Sheets("Main Logic").Range("CLLIQuad")

' --expand name definition to include current number of rows in formula range
  ActiveWorkbook.Names.Add Name:="CLLIQuad", RefersTo:="='Main Logic'!" & QuadRange.
Resize(rowsize:=formulaRange.rows.Count).Address

' --remove reference
  Set QuadRange = Nothing

End Sub

Public Sub BuildFixedCostTable(formulaRange As Range)
Dim WireCenter          As String, _
    HoldWireCenter      As String
Dim ws                  As Worksheet
Dim FixedCostTable      As Range, _
    Cell                As Range
Dim rowNumber           As Long, _
    LastRow             As Long, _
    FirstRow            As Long, _
    TableIndex          As Long, _
    FormulaColumn       As Long
Dim FirstRecord         As Boolean

  Set ws = ActiveWorkbook.Sheets("Main Logic")
  Set FixedCostTable = ws.Range("FixedCostTable")

  With FixedCostTable
    .ClearContents
    .Resize rowsize:=1
  End With

  FirstRecord = True
  FormulaColumn = ws.Range("Swlinecost").columns(1).Column
  TableIndex = 1

  For Each Cell In formulaRange.columns(1).rows
    rowNumber = Cell.row
    WireCenter = ws.Cells(rowNumber, 1).Text
    If WireCenter <> HoldWireCenter Then
      If FirstRecord Then
        FirstRecord = False
        FirstRow = rowNumber
        HoldWireCenter = WireCenter
      Else
'       --pending wire center ended on previous row
        LastRow = rowNumber - 1
        With FixedCostTable
          .Resize rowsize:=TableIndex
          .Cells(TableIndex, 1).Formula = "=r" & FirstRow & "c1"
          .Cells(TableIndex, 2).Formula = "=sum(r" & FirstRow & "c" & FormulaColumn
& ":r" & LastRow & "c" & FormulaColumn & ")"
          .Cells(TableIndex, 3).Formula = "=IF(rc[-1]=0,0,((VLOOKUP(r" & FirstRow &
"c3,COSwitchCost,2,FALSE)*TrfSen)/rc[-1]*SwitchFillFactor)*(1+(VLOOKUP(r" & FirstRow
```

```
 & "c3,COSwitchCost,4,FALSE)+VLOOKUP(r" & FirstRow & "c3,COSwitchCost,5,FALSE)))))"
          .Cells(TableIndex, 4).Formula = "=(VLOOKUP(r" & FirstRow & "c3,COSwitchCos
t,3,FALSE)/SwitchFillFactor*TrfSen)*(1+VLOOKUP(r" & FirstRow & "C3,COSwitchCost,4,FA
LSE)+VLOOKUP(r" & FirstRow & "C3,COSwitchCost,5,FALSE))"
          .Cells(TableIndex, 5).Formula = "=sum(rc[-2]:rc[-1])"
        End With
        HoldWireCenter = WireCenter
        FirstRow = rowNumber
        TableIndex = TableIndex + 1
      End If
    End If
  Next Cell


' --write last table entry, rowNumber now takes place of lastRow
  With FixedCostTable
    .Resize rowsize:=TableIndex
    .Cells(TableIndex, 1).Formula = "=r" & FirstRow & "c1"
    .Cells(TableIndex, 2).Formula = "=sum(r" & FirstRow & "c" & FormulaColumn & ":r"
 & rowNumber & "c" & FormulaColumn & ")"
    .Cells(TableIndex, 3).Formula = "=IF(rc[-1]=0,0,((VLOOKUP(r" & FirstRow & "c3,CO
SwitchCost,2,FALSE)*TrfSen)/rc[-1]*SwitchFillFactor)*(1+(VLOOKUP(r" & FirstRow & "c3
,COSwitchCost,4,FALSE)+VLOOKUP(r" & FirstRow & "c3,COSwitchCost,5,FALSE))))"
    .Cells(TableIndex, 4).Formula = "=(VLOOKUP(r" & FirstRow & "c3,COSwitchCost,3,FA
LSE)/SwitchFillFactor*TrfSen)*(1+VLOOKUP(r" & FirstRow & "C3,COSwitchCost,4,FALSE)+V
LOOKUP(r" & FirstRow & "C3,COSwitchCost,5,FALSE))"
    .Cells(TableIndex, 5).Formula = "=sum(rc[-2]:rc[-1])"
  End With

' --adjust range name
  ActiveWorkbook.Names.Add Name:="FixedCostTable", RefersTo:="='Main Logic'!" & Fixe
dCostTable.Resize(rowsize:=TableIndex).Address

End Sub

Public Function GetWireCenterData(FileName As String, Targets As Variant) As Variant
Dim Records()           As String, _
    IndexFile           As String
Dim stream              As Integer, _
    CLLI_Length         As Integer
Dim RecordPosition      As Long, _
    RecordCount         As Long, _
    index               As Long
Dim WireCenter          As String, _
    buffer              As String, _
    TargetWireCenter    As String

  IndexFile = ExtractPath(FileName) & ExtractName(FileName, False) & ".idx"

  For index = 1 To UBound(Targets)
    TargetWireCenter = Targets(index)
    RecordPosition = GetPrivateProfileInt("Wire Centers", TargetWireCenter, -1, Inde
xFile)

    If RecordPosition > 0 Then
      stream = FreeFile()
      Open FileName For Input Shared As #stream

      CLLI_Length = ThisWorkbook.Sheets("RangeNames").Range("CLLI_Name_Length").Valu
e
```

```
      WireCenter = TargetWireCenter

      Seek #stream, RecordPosition

      Do Until EOF(stream) Or (WireCenter <> TargetWireCenter)
        Line Input #stream, buffer
        If Len(buffer) > CLLI_Length Then
          WireCenter = Left$(buffer, CLLI_Length)
          If WireCenter = TargetWireCenter Then
            RecordCount = RecordCount + 1
            ReDim Preserve Records(RecordCount)
            Records(RecordCount) = buffer
          End If
        End If
      Loop

      Close stream
    End If
  Next index

  GetWireCenterData = Records

End Function

Public Sub ReviewState(ByVal StateIndex As Long)
Dim OutputFile              As String, _
    FileSpec                As String, _
    IndexFile               As String
Dim index                   As Long, _
    wcIndex                 As Long
Dim WireCenters             As Variant
Dim Records                 As Variant
Dim List                    As Listbox

  Set List = ThisWorkbook.Sheets("Review").ListBoxes("lstWireCenters")

  FileSpec = ThisWorkbook.Sheets("RangeNames").Range("Input_Path") & CurrentModel.CB
GDataSet

' --return if there are no input files
  If Not FileExists(FileSpec) Then
    ShowStatusMessage FileSpec & " does not exist ...."
    Exit Sub
  End If

' --build index file
  IndexFile = BuildFileIndexes(FileSpec)

' --build wire center array
  ReDim WireCenters(1)
  For index = 1 To List.ListCount
    If List.Selected(index) Then
      wcIndex = wcIndex + 1
      ReDim Preserve WireCenters(wcIndex)
      WireCenters(wcIndex) = List.List(index)
    End If
  Next index

'   --Records is an array of all wire center records
```

```
  ClearWireCenterData
  ShowStatusMessage "retrieving data for selected wire centers"
  Records = GetWireCenterData(FileSpec, WireCenters)
  ShowStatusMessage "moving wire center data to model"
  MoveWireCenterData Records
  ShowStatusMessage "copying formulas, adjusting fixed cost table"
  ProcessWireCenterData UBound(Records), True
  ShowStatusMessage

End Sub

Public Sub ProcessState(ByVal StateIndex As Long)
Dim OutputFile              As String, _
    FileSpec                As String, _
    IndexFile               As String, _
    StateName               As String
Dim index                   As Long, _
    stream                  As Long
Dim WireCenters             As Variant, _
    WireCenter              As Variant
Dim EndTime                 As Variant, _
    StartTime               As Variant, _
    ElapsedTime             As Variant, _
    Records                 As Variant

  FileSpec = ThisWorkbook.Sheets("RangeNames").Range("Input_Path") & CurrentModel.CB
GDataSet
  StateName = ThisWorkbook.Sheets("RangeNames").Range("StateList").Offset(StateIndex
 - 1, 0).Cells(1, 1).Value

' --return if there are no input files
  If Not FileExists(FileSpec) Then
    PostStatusMessage StateName, "--", 0, FileSpec & " does not exist"
    ShowMessage FileSpec & " does not exist ...."
    Exit Sub
  End If

' --open "cbg" output file
  stream = FreeFile()
  OutputFile = ThisWorkbook.Worksheets("RangeNames").Range("Output_Path") & CurrentM
odel.Name & ".csv"
  Open OutputFile For Output As #stream


' --build index file
  IndexFile = BuildFileIndexes(FileSpec)

' --build wire center array
  WireCenters = GetAllSettings("Wire Centers", IndexFile)
  ReDim WireCenter(1)

  StartTime = Now()

' --process all files
  For index = 1 To UBound(WireCenters)
    If EndTime > StartTime Then
      ShowStatusMessage "Elapsed Time: " & Format$(EndTime - StartTime, "hh.nn.ss")
& "    " & WireCenters(index)
    Else
      ShowStatusMessage "processing " & WireCenters(index) & " ......"
```

```
    End If
'   --Records is an array of all wire center records
    WireCenter(1) = WireCenters(index)
    Records = GetWireCenterData(FileSpec, WireCenter)
    MoveWireCenterData Records
    ProcessWireCenterData UBound(Records), False
    WriteWireCenterData stream
    ClearWireCenterData
    EndTime = Now()
  Next index

  ElapsedTime = Now() - StartTime
  PostStatusMessage StateName, "OK", ElapsedTime, ""

' --close output stream
  Close stream

  IndexFile = BuildFileIndexes(OutputFile, 48)

End Sub

Public Sub PostStatusMessage(StateName As String, Status As String, ElapsedTime As V
ariant, Msg As String)
Dim rngData           As Range

  Set rngData = ThisWorkbook.Sheets("Process Status").Range("StatusDataArea")

  rngData.Offset(StatusReportLine, 0).Cells(1, 1).Value = StateName
  rngData.Offset(StatusReportLine, 0).Cells(1, 2).Value = Status
  rngData.Offset(StatusReportLine, 0).Cells(1, 3).Value = Format$(ElapsedTime, "hh.n
n.ss")
  rngData.Offset(StatusReportLine, 0).Cells(1, 4).Value = Msg

  StatusReportLine = StatusReportLine + 1

  Set rngData = Nothing

End Sub

Public Sub ClearWireCenterData()
Dim ws        As Worksheet
Dim rng       As Range

' --clear "Main Logic"
  On Local Error Resume Next

  Set ws = ActiveWorkbook.Sheets("Main Logic")
  ws.Activate
  Set rng = ws.Cells(1, 1).CurrentRegion
  rng.Offset(2, 0).Resize(rng.rows.Count - 2).ClearContents
  Set ws = ActiveWorkbook.Sheets("Outputs")
  ws.Activate
  Set rng = ws.Cells(1, 1).CurrentRegion
  rng.Offset(3, 0).Resize(rng.rows.Count - 2).ClearContents

  Set ws = Nothing
  Set rng = Nothing

End Sub
```

```
Public Sub MoveWireCenterData(Records As Variant)
Const MAXRECORDS = 250
Dim ValuesArray            As Variant
Dim index                  As Long, _
    RecordCount            As Long, _
    row                    As Long, _
    MaxColumns             As Long, _
    Count                  As Long, _
    RecordsLoaded          As Long, _
    FirstRow               As Long, _
    LastRow                As Long, _
    RemainingRecords       As Long

• --activate data sheet
  ActiveWorkbook.Sheets("Main Logic").Activate

• --initialize variables
  row = ThisWorkbook.Sheets("RangeNames").Range("Starting_DataRow").Value
  MaxColumns = ThisWorkbook.Sheets("RangeNames").Range("Max_Input_Columns").Value
  RecordCount = UBound(Records)

• --allocate space for data fields
  ReDim ValuesArray(MAXRECORDS)

• --load 200 records at a time
  Count = 1
  FirstRow = row

  Do While Count * MAXRECORDS <= RecordCount
    For index = 1 To MAXRECORDS
      RecordsLoaded = RecordsLoaded + 1
      ValuesArray(index) = ParseRecord(Records(RecordsLoaded))
    Next index
    LastRow = row + (Count * MAXRECORDS) - 1
    Range(Cells(FirstRow, 1), Cells(LastRow, MaxColumns)).Value = ValuesArray
    FirstRow = LastRow + 1
    Count = Count + 1
  Loop

  RemainingRecords = RecordCount - RecordsLoaded
  ReDim ValuesArray(RemainingRecords)

  For index = 1 To RemainingRecords
    RecordsLoaded = RecordsLoaded + 1
    ValuesArray(index) = ParseRecord(Records(RecordsLoaded))
  Next index

  Range(Cells(FirstRow, 1), Cells(FirstRow + RemainingRecords - 1, MaxColumns)).Valu
e = ValuesArray

End Sub

Public Function ParseRecord(ByVal buffer As Variant, Optional columns As Variant) As
 Variant
Dim Pos1                   As Long, _
    Pos2                   As Long, _
    index                  As Long, _
    MaxColumns             As Long

    On Error GoTo ParseRecord_Error
```

```
    If IsMissing(columns) Then
      MaxColumns = ThisWorkbook.Sheets("RangeNames").Range("Max_Input_Columns").Valu
e
    Else
      MaxColumns = columns
    End If

    If Len(Trim(buffer)) = 0 Then
        ParseRecord = False
        Exit Function
    End If

    ReDim Values(MaxColumns)

    Pos1 = 1
    Pos2 = InStr(Pos1, buffer, ",")
    index = 0

    While Pos2
        index = index + 1
        Values(index) = Trim$(Mid(buffer, Pos1, Pos2 - Pos1))
        Pos1 = Pos2 + 1
        Pos2 = InStr(Pos1, buffer, ",")
    Wend

    Values(index + 1) = Trim$(Mid(buffer, Pos1))

    ParseRecord = Values
    Exit Function

ParseRecord_Error:

    Select Case Err
        Case 9
            ShowMessage "Message: Error in ParseRecord ..." & p_CRLF _
                      & "Error: " & Error() & p_CRLF _
                      & "Action:  Terminating processing"
            EndApplication
    End Select

End Function

Public Sub ProcessWireCenterData(NumberOfRows As Long, Optional MultiWireCenter As V
ariant)
Dim tempRange           As Range, _
    formulaRange        As Range, _
    outRange            As Range
Dim ws                  As Worksheet, _
    wsOut               As Worksheet
Dim LastRow             As Long, _
    FirstRow            As Long, _
    TableIndex          As Long, _
    rowNumber           As Long
Dim MultiWCFlag         As Boolean

  If IsMissing(MultiWireCenter) Then
    MultiWCFlag = False
  Else
    MultiWCFlag = MultiWireCenter
```

```
  End If

' -- copy formulas, write results
  Set ws = ActiveWorkbook.Worksheets("Main Logic")
  ws.Activate

' --define area that calculations will be copied to
  Set tempRange = ws.Range("MainCalculations")
  Set formulaRange = ws.Range("MainCalculations").Resize(rowsize:=NumberOfRows)

' --define boundary rows for sorting
  FirstRow = formulaRange.row
  LastRow = FirstRow + NumberOfRows - 1

  If NumberOfRows > 1 Then
'   --copy formulas if more than one row
    formulaRange.FillDown
'   --sort thru 'Segment B' calculation
    SortCalculations FirstRow, LastRow
  End If

' --build entry in FixedCost Table
  If MultiWCFlag Then BuildFixedCostTable formulaRange

  SetCLLIQuadRange formulaRange

  Calculate

' --copy formulas in 'outputs' sheet
  Set wsOut = ActiveWorkbook.Worksheets("Outputs")
  wsOut.Activate

  Set outRange = wsOut.Range("OutputCalculations").Resize(rowsize:=NumberOfRows)
  If NumberOfRows > 1 Then
    outRange.FillDown
  End If

' --recalculate sheets
  Calculate

' --erase objecs
  Set tempRange = Nothing
  Set formulaRange = Nothing
  Set outRange = Nothing
  Set ws = Nothing
  Set wsOut = Nothing

End Sub

Public Sub SortCalculations(StartRow As Long, EndRow As Long)
Dim SortRange     As Range

  Set SortRange = Range(Cells(StartRow, 1), Cells(EndRow, 23))

  With SortRange
    .Select
    .Activate
    .Calculate
  End With
```

```
  Selection.Sort Key1:=Range("A3"), Order1:=xlAscending, _
                 Key2:=Range("G3"), Order2:=xlAscending, _
                 Key3:=Range("W3"), Order3:=xlAscending, _
                 Header:=xlGuess, OrderCustom:=1, matchcase:=False, _
                 Orientation:=xlTopToBottom

  Set SortRange = Nothing

End Sub

Public Sub WriteWireCenterData(stream As Long)
Dim outputRange     As Range, _
    cr              As Range
Dim ws              As Worksheet
Dim OutputString    As String
Dim rw              As Range, _
    Cell            As Range
Dim CellContents    As Variant
Dim TotalRow        As Long

    On Error GoTo WriteWireCenterData_Error

    Set ws = ActiveWorkbook.Sheets("Outputs")
    Set cr = ws.Cells(1, 1).CurrentRegion

    Set outputRange = cr.Offset(2, 0).Resize(cr.rows.Count - 2, cr.columns.Count)

    For Each rw In outputRange.rows
        OutputString = rw.Cells(1, 1).Value
        For Each Cell In rw.Cells
            CellContents = Cell.Cells(1, 1).Value
'           --this should only be temporary until calculations are firmed up
            If IsNumeric(CellContents) Then
                CellContents = Format(CellContents, "#.00")
            End If
            If Cell.Column > 1 Then
              If Cell.Column <> 6 Then
                OutputString = OutputString & "," & CStr(CellContents)
              Else
                OutputString = OutputString & "," & Format$(CellContents, "##")
              End If
            End If
        Next Cell
        Print #stream, OutputString
    Next rw

    Exit Sub

WriteWireCenterData_Error:

    ShowMessage "Message: Error writing output file ...." & _
                p_CRLF & "Action: Terminating application" & _
                p_CRLF & "Choose Window/Model on main Excel menu to investigate"
    Windows("Control").Activate
    EndApplication

End Sub
```

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

' --private constants
  Private Const Process_ID = 1
  Private Const Review_ID = 2
  Private Const Reports_ID = 4
  Private Const Workbooks_ID = 8
  Private Const Instructions_ID = 16
  Private Const InputTables_ID = 32
  Private Const Options_ID = 64

' --private variables
  Private MoveControlsOnSheet      As Long
  Private ModelWB                  As Workbook

' --public variables
  Public p_CRLF                   As String
  Public p_ReturningFromWB        As Boolean
  Public p_ReturningFromOptions   As Boolean
  Public StatusReportLine         As Long

' --public arrays
  Public p_FileNames()            As String


'=================================================================================
============
Public Sub aaaaa()
  Sheets("options").Cells(1, 1).RowHeight = Sheets("Process").Cells(1, 1).RowHeight
  Sheets("options").Cells(1, 1).ColumnWidth = Sheets("Process").Cells(1, 1).ColumnWi
dth

End Sub

Public Sub Auto_Close()

  CloseControlWorkbook

End Sub

'=================================================================================
============
Public Sub Auto_Open()
Dim obj             As Object

  CheckLicenseAgreement
  CreateMenus
  CloseAllOpenWorkbooks

  MoveControlsOnSheet = Process_ID + Review_ID + Reports_ID + Instructions_ID + Inpu
tTables_ID + Options_ID

  With ThisWorkbook
    .Sheets("Process").OnSheetActivate = "ProcessSheet_GotFocus"
    .Sheets("Review").OnSheetActivate = "ReviewSheet_GotFocus"
    .Sheets("Reports").OnSheetActivate = "ReportsSheet_GotFocus"
    .Sheets("Instructions").OnSheetActivate = "InstructionsSheet_GotFocus"
    .Sheets("Input Tables").OnSheetActivate = "InputTablesSheet_GotFocus"
    .Sheets("Input Tables").OnSheetDeactivate = "InputTablesSheet_LostFocus"
    .Sheets("Options").OnSheetActivate = "OptionsSheet_GotFocus"
  End With
```

```
  With ActiveWindow
    .WindowState = xlMaximized
    .Caption = "Control"
  End With

  With Application
    .Caption = Empty
    .Caption = "BCPM Model"
    .Calculation = xlManual
  End With

  ThisWorkbook.Sheets("Instructions").Activate

  HideSheets
  SetPaths
  UpdateModelList

End Sub

Public Sub HideSheets()
Dim obj         As Object

' --hide all modules
  For Each obj In ThisWorkbook.Modules
    obj.Visible = False
  Next obj

  For Each obj In ThisWorkbook.DialogSheets
    obj.Visible = False
  Next obj

  ThisWorkbook.Sheets("RangeNames").Visible = False
  ThisWorkbook.Sheets("Aggregate Support Calculations").Visible = False

End Sub

Public Sub BailOut()
Dim obj         As Object

' --hide all modules
  For Each obj In ThisWorkbook.Modules
    obj.Visible = True
  Next obj

  For Each obj In ThisWorkbook.DialogSheets
    obj.Visible = True
  Next obj

  ThisWorkbook.Sheets("RangeNames").Visible = True
  ThisWorkbook.Sheets("Aggregate Support Calculations").Visible = True

End Sub

' ================================================================
' GotFocus events for the various sheets
' ================================================================
Public Sub OptionsSheet_GotFocus()
Dim sht       As Worksheet
Dim obj       As Object
```

```
  On Local Error Resume Next

  Set sht = ThisWorkbook.Sheets("Options")

  sht.Unprotect

  If MoveControlsOnSheet And Options_ID Then
    MoveControlsOnSheet = MoveControlsOnSheet - Options_ID
    HideThings sht
  End If

  sht.Protect

  Set sht = Nothing
  Set obj = Nothing

End Sub

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

  On Local Error Resume Next

  Set sht = ThisWorkbook.Sheets("Process")

  With sht
    .Move before:=Sheets(1)
    .Unprotect
  End With

  If p_ReturningFromWB Then
    p_ReturningFromWB = False
    Exit Sub
  End If

  If MoveControlsOnSheet And Process_ID Then
    MoveControlsOnSheet = MoveControlsOnSheet - Process_ID
    HideThings sht
  End If

  For Each obj In sht.TextBoxes
    obj.Text = ""
    SmallAndBlue obj
    If obj.Name = "tbTime" Then
      obj.Visible = False
    End If
  Next obj

  For Each obj In sht.DropDowns
    If obj.ListCount Then
      obj.ListIndex = 0
    End If
  Next obj

  ClearStateSelections

  UpdateModelList

  sht.Protect
```