```
  Set sht = Nothing
  Set obj = Nothing

End Sub


Public Sub InstructionsSheet_GotFocus()

  ActiveWindow.DisplayVerticalScrollBar = True

End Sub


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

  On Local Error Resume Next

  Set sht = ThisWorkbook.Sheets("Review")

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

  If p_ReturningFromWB Then
    p_ReturningFromWB = False
    Exit Sub
  End If

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

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

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

  sht.ListBoxes("lstWireCenters").RemoveAllItems

  ClearStateSelections

  UpdateModelList

  sht.Protect

  Set sht = Nothing
  Set obj = Nothing

End Sub

Public Sub ReportsSheet_GotFocus()
Dim sht          As Worksheet
```

```
Dim dd          As DropDown
Dim FileSpec    As String

  On Local Error Resume Next

  Set sht = ThisWorkbook.Sheets("Reports")

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

  If p_ReturningFromWB Then
    p_ReturningFromWB = False
    Exit Sub
  End If

  If p_ReturningFromOptions Then
    p_ReturningFromOptions = False
    Exit Sub
  End If

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

  ClearReportsSheet

  Set dd = ThisWorkbook.Sheets("Reports").DropDowns("FactorSets")
  dd.RemoveAllItems
  FileSpec = ThisWorkbook.Sheets("RangeNames").Range("Factor_Path") & "*.inf"
  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"
  End If

  dd.List = p_FileNames

  SmallAndBlue sht.TextBoxes("txtPrimaryKey")

  UpdateModelList

  sht.Protect

  Set sht = Nothing

End Sub

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

  If p_ReturningFromWB Then
    Exit Sub
  End If

  Set sht = ThisWorkbook.Sheets("Input Tables")

  With sht
```

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

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

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

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

  sht.Protect

End Sub

Public Sub InputTablesSheet_LostFocus()

    p_ReturningFromWB = False

End Sub

Public Sub SmallAndBlue(tb As Textbox)

  'ActiveSheet.DrawingObjects("tb").Select
  With tb.Font
    .Name = "MS Sans Serif"
    .FontStyle = "Regular"
    .Size = 8.5
    .Strikethrough = False
    .Superscript = False
    .Subscript = False
    .OutlineFont = False
    .Shadow = False
    .Underline = xlNone
    .ColorIndex = 5
  End With

End Sub

'=================================================================================
============
Public Sub btnProcessOK_Click()
Dim index            As Long, _
    WorkingIndex     As Long
Dim StartTime        As Variant, _
    EndTime          As Variant, _
    ElapsedTime      As Variant, _
    States           As Variant
Dim ws               As Worksheet
Dim dd               As DropDown
Dim tbTime           As Textbox
Dim rng              As Range
Dim RngRMF           As Range

  Set ws = ThisWorkbook.Sheets("Process")
```

```
  Set tbTime = ws.TextBoxes("tbTime")

' --make sure ElapsedTime box is visible
  tbTime.Visible = True

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

  ShowMessage

' --make sure state & model have been selected
  If Not CheckProcessSelections Then
    Exit Sub
  End If

  Application.ScreenUpdating = False

' --loop through all states
  If LoadBCPMModel() Then
'   --initialize status report
    PostModelInfoOnStatusReport
    Application.Goto ModelWB.Sheets("Main Logic").Cells(1, 1)
    States = SetSelectedStates(ThisWorkbook.Sheets("Process").ListBoxes("lstStates")
)
    Set rng = ThisWorkbook.Sheets("RangeNames").Range("StateList")
    StartTime = Now()
    For index = 1 To UBound(States)
      Set RngRMF = ModelWB.Sheets("Misc Table Inputs").Range("ResMultiTable")
      ModelWB.Sheets("Fixed Tables").Range("ResLinesMultiplier").Value = RngRMF.Offs
et(States(index) - 1, 3).Cells(1, 1).Value
      WorkingIndex = States(index)
      ShowMessage "Processing " & rng.Offset(WorkingIndex - 1, 0).Cells(1, 1) & " ..
.."
      SetPaths rng.Offset(WorkingIndex - 1, -1).Cells(1, 1).Value
      ProcessState WorkingIndex
      EndTime = Now()
      ElapsedTime = EndTime - StartTime
      tbTime.Text = "Cumulative Processing Time: " & Format$(ElapsedTime, "hh.nn.ss"
)
    Next index
'   --close the model
    CloseBCPModel
  Else
    ShowMessage "Unable to load model " & CurrentModel.Name
  End If

  ShowStatusMessage

  Set ws = Nothing
  Set tbTime = Nothing
  Set rng = Nothing

End Sub

Public Sub PostModelInfoOnStatusReport()
Dim rngModel            As Range
Dim rngData             As Range
```

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

  rngModel.ClearContents
  rngData.ClearContents

  rngModel.Cells(1, 1).Value = CurrentModel.Name
  rngModel.Cells(2, 1).Value = LCase$(CurrentModel.LogicSet)
  rngModel.Cells(3, 1).Value = CurrentModel.CBGDataSet

  StatusReportLine = 0

  Set rngModel = Nothing
  Set rngData = Nothing

End Sub

Public Sub ReportsModels_Click()

  PostModelInformation

End Sub

Public Sub ReviewModels_Click()

  PostModelInformation
  GetWireCenters

End Sub

Public Sub btnReviewOK_Click()
Dim index           As Long
Dim rng             As Range
Dim dd              As DropDown
Dim RngRMF          As Range

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

  p_ReturningFromWB = True

  Application.ScreenUpdating = False

' --loop through all states
  If LoadBCPMModel() Then
    Workbooks(CurrentModel.LogicSet).Sheets("Main Logic").Activate
    Set rng = ThisWorkbook.Sheets("RangeNames").Range("StateList")
    index = ThisWorkbook.Sheets("Review").ListBoxes("lstStates").ListIndex

    Set RngRMF = ModelWB.Sheets("Misc Table Inputs").Range("ResMultiTable")
    ModelWB.Sheets("Fixed Tables").Range("ResLinesMultiplier").Value = RngRMF.Offset
(index - 1, 3).Cells(1, 1).Value

    SetPaths rng.Offset(index - 1, -1).Cells(1, 1).Value
    ReviewState index
'   --put in descriptions for report
```

```
    With ActiveWorkbook.Sheets("Report")
      .Range("State") = "State: " & rng.Offset(index - 1, 0).Cells(1, 1).Value
      .Range("InputData") = "Input Data: " & CurrentModel.Name & ".csv"
      .Range("Workbook") = "Workbook: " & CurrentModel.LogicSet
      .Range("Description") = "Description: " & CurrentModel.Description
      .Calculate
    End With
'   --activate model
    Application.Goto reference:=ActiveWorkbook.Sheets("Main Logic").Range("a2")
  Else
    ShowStatusMessage "Unable to load model " & CurrentModel.LogicSet
  End If

End Sub

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

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

  If List.ListCount Then
    ReDim Choices(List.ListCount)
    List.Selected = Choices
  End If

  Set List = Nothing

End Sub


'==================================================================================
============
' CheckSelections:  Checks to make sure that at least one state is selected, and als
o a model has
'                   been chosen.  Displays message for first thing that is not selec
ted.
'
Private Function CheckProcessSelections() As Boolean
Dim rtn          As Variant
Dim List         As Listbox

  ShowMessage

' --reset range names
  ThisWorkbook.Sheets("RangeNames").Range("Curr_StateName").Value = ""
  ThisWorkbook.Sheets("RangeNames").Range("Curr_StateIndex").Value = -1

' --check states first
  Set List = ThisWorkbook.Sheets("Process").ListBoxes("lstStates")
  rtn = SetSelectedStates(List)
  CheckProcessSelections = False

  If rtn(1) = -1 Then
    ShowMessage "Must select one or more states"
    Exit Function
  End If

' --check model selection
  If ThisWorkbook.Sheets("Process").DropDowns("ddModels").ListIndex = 0 Then
```

```
    ShowMessage "Must select a model"
    Exit Function
  End If

  CheckProcessSelections = True

  Set List = Nothing

End Function

'==================================================================================
============
Public Sub CloseAllOpenWorkbooks()
Dim tempWorkbook        As Workbook
Dim SaveFileName        As String

  ShowMessage "closing all open workbooks ..."

  For Each tempWorkbook In Workbooks
    If tempWorkbook.Name <> ThisWorkbook.Name Then
      SaveWorkbook tempWorkbook
      ForceCloseWorkbook tempWorkbook
    End If
  Next tempWorkbook

  ShowMessage

End Sub

'==================================================================================
============
Public Sub CloseBCPModel()

' --make sure the model is the active workbook
  On Local Error Resume Next
  ModelWB.Activate
  If Err = 0 Then
    ActiveWorkbook.Saved = True
    SaveWorkbook ActiveWorkbook
    ForceCloseWorkbook ActiveWorkbook
    Set ModelWB = Nothing
  Else
    ShowStatusMessage "Benchmark Cost Model not currently active"
    Err = 0
  End If

End Sub

'==================================================================================
============
  Public Sub CloseControlWorkbook()

    RemoveMenus

    With Application
      .Caption = Empty
    End With

    With ThisWorkbook
      .OnSheetActivate = ""
      .OnSheetDeactivate = ""
```

```
      .Saved = True
      .Close
    End With

  End Sub

' ==============================================================================
==============
  Public Sub DoNothing()

    MsgBox "Under development ...", , "BCPM Message"

  End Sub

' ==============================================================================
==============
  Public Sub EndApplication()

    Windows("Control").Activate
    ShowStatusMessage
    End

  End Sub

' ==============================================================================
==============
Public Function FileExists(FileName As String) As Boolean
Dim stream            As Long

  stream = FreeFile()

  On Error Resume Next
  Open FileName For Input Shared As #stream

  FileExists = (Err = 0)

  Close stream

End Function

' ==============================================================================
==============
Public Sub ForceCloseWorkbook(someWorkbook As Workbook)

  With someWorkbook
    .Saved = True
    .Close
  End With

End Sub

' ==============================================================================
==============
Public Sub FormatInputReportsDialog()
Dim lst       As Listbox
Dim ws        As Worksheet

  Set lst = ThisWorkbook.DialogSheets("dlgInputReports").ListBoxes("Sheets")

  lst.RemoveAllItems
```

```
  For Each ws In ActiveWorkbook.Worksheets
    lst.AddItem ws.Name
  Next ws

End Sub

' ==============================================================================
==============
  Public Function GetFileNames(FileSpec As String) As Boolean
  Dim FileCount        As Integer
  Dim FoundFile        As String

'   --initialize variables
    FileCount = 0
    FoundFile = Dir(FileSpec)

'   --exit if no files are found
    If Len(FoundFile) = 0 Then
'     --show message indicating no input files for selected state
      GetFileNames = False
      Exit Function
    End If

'   --now look for rest of files
    Do While Len(FoundFile) > 0
      FileCount = FileCount + 1
      ReDim Preserve p_FileNames(FileCount) As String
'     !!warning: path must be added in calling routine
      p_FileNames(FileCount) = FoundFile
      FoundFile = Dir()
    Loop

'   --return success
    GetFileNames = True

  End Function

' ==============================================================================
==============
Public Function SetSelectedStates(anyList As Listbox) As Variant
Dim index            As Long, _
    RtnValue         As Long, _
    StateIndex       As Long
Dim Array()          As Long

  With ThisWorkbook.Sheets("RangeNames")
    .Range("Curr_StateIndex") = -1
    .Range("Curr_StateName") = ""
  End With

' --set defaults, assume no state selected
  ReDim Array(1)
  Array(1) = -1

  If anyList.MultiSelect = xlNone Then
'   --list is single select
    RtnValue = anyList.ListIndex
    Array(1) = RtnValue
    With ThisWorkbook.Sheets("RangeNames")
      .Range("Curr_StateIndex") = RtnValue
```

```
      .Range("Curr_StateName") = anyList.Value
      .Range("Selected_States") = RtnValue
    End With
  Else
'   --list supports multiple selection
    For index = 1 To anyList.ListCount
      If anyList.Selected(index) Then
        StateIndex = StateIndex + 1
        ReDim Preserve Array(StateIndex)
        Array(StateIndex) = index
      End If
    Next index
  End If

  SetSelectedStates = Array

End Function

' ==============================================================================
=============
Private Function LoadBCPMModel() As Boolean
Dim Model            As String
Dim wb               As Workbook

    On Local Error GoTo LoadBCPMModel_Error

    Model = ThisWorkbook.Sheets("RangeNames").Range("Model_WB_Name")

'   --check to make sure model isn't already loaded
    For Each wb In Workbooks
      If wb.FullName = Model Then
        LoadBCPMModel = True
        Set ModelWB = wb
        Exit Function
      End If
    Next wb

    ShowStatusMessage "retrieving " & LCase(Model) & " ...."

    Workbooks.Open FileName:=Model
    Set ModelWB = ActiveWorkbook

    ShowStatusMessage

    LoadBCPMModel = True
    Exit Function

LoadBCPMModel_Error:

    LoadBCPMModel = False
    Exit Function

End Function

Public Sub LoadCapCostWorkbook()
Dim Model            As String
Dim wb               As Workbook

    On Local Error GoTo LoadCapCostWorkbook_Error

    Model = ThisWorkbook.Path & "\factors\capcost.xls"
```

```
•   --check to make sure model isn't already loaded
    For Each wb In Workbooks
      If wb.FullName = Model Then
        wb.Activate
        Exit Sub
      End If
    Next wb

    ShowStatusMessage "retrieving " & LCase(Model) & " ...."

    Workbooks.Open FileName:=Model

    ShowStatusMessage

    Exit Sub

LoadCapCostWorkbook_Error:

    Exit Sub

End Sub

Public Function LoadReportWorkbook() As Boolean
Dim Model           As String
Dim wb              As Workbook

    On Local Error GoTo LoadReportWorkbook_Error

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

•   --check to make sure model isn't already loaded
    For Each wb In Workbooks
      If wb.FullName = Model Then
        wb.Activate
        LoadReportWorkbook = True
        Exit Function
      End If
    Next wb

    ShowStatusMessage "retrieving " & LCase(Model) & " ...."

    Workbooks.Open FileName:=Model
    ActiveWindow.Caption = "Reports"

    ShowStatusMessage

    LoadReportWorkbook = True

    Exit Function

LoadReportWorkbook_Error:

    LoadReportWorkbook = False
    Exit Function

End Function

'==========================================================================
============
Public Sub GetWireCenters()
```

```
Dim IndexFile            As String, _
    FileSpec             As String
Dim index                As Long
Dim dd                   As DropDown
Dim List                 As Listbox
Dim rng                  As Range

  Set rng = ThisWorkbook.Sheets("RangeNames").Range("StateList")
  Set dd = ThisWorkbook.Sheets("Review").DropDowns("ddModels")
  Set List = ThisWorkbook.Sheets("Review").ListBoxes("lstWireCenters")

  List.RemoveAllItems

  index = ThisWorkbook.Sheets("Review").ListBoxes("lstStates").ListIndex
  If index Then
    SetPaths rng.Offset(index - 1, -1).Cells(1, 1).Value
    If dd.ListIndex > 0 Then
      SetCurrentModel dd.List(dd.ListIndex)
    Else
      ThisWorkbook.Sheets("Review").ListBoxes("lstStates").ListIndex = 0
      MsgBox "You must select a model"
      Exit Sub
    End If
    FileSpec = ThisWorkbook.Sheets("RangeNames").Range("Input_Path") & CurrentModel.
CBGDataSet
    If FileExists(FileSpec) Then
      LoadWireCenters ThisWorkbook.Sheets("Review").ListBoxes("lstWireCenters")
    End If
  End If

  Set rng = Nothing
  Set dd = Nothing

End Sub

Public Sub LoadWireCenters(lstAny As Listbox)
Dim stream               As Long, _
    index                As Long
Dim WireCenter           As String, _
    buffer               As String, _
    IndexFile            As String
Dim FoundSection         As Boolean
Dim WireCenters()        As Variant

    lstAny.RemoveAllItems

    buffer = ThisWorkbook.Sheets("RangeNames").Range("Input_Path") & CurrentModel.CB
GDataSet
    IndexFile = SetIndexFile(buffer)

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

      --initialize variables
      FoundSection = False

      ShowStatusMessage "Loading Wire Centers ...."

      Do Until EOF(stream)
```

```
        Line Input #stream, buffer
        If Len(buffer) Then
          If FoundSection Then
            --make sure not invading another section
            If InStr(buffer, "[") = 0 Then
              index = index + 1
              ReDim Preserve WireCenters(index)
              WireCenters(index) = Left$(buffer, InStr(buffer, "=") - 1)
            Else
              Exit Do
            End If
          Else
            If InStr(buffer, "Wire Centers") Then
              FoundSection = True
            End If
          End If
        End If
      Loop

      SelectionSort WireCenters

      lstAny.List = WireCenters

      Close stream

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

End Sub

Private Function SetIndexFile(FileName As String) As String
Dim position        As Long
Dim IndexFile       As String

  position = InStr(FileName, ".")
  IndexFile = Left$(FileName, position) & "idx"

  If Not FileExists(IndexFile) Then
    BuildFileIndexes FileName
  End If

  SetIndexFile = IndexFile

End Function

'==================================================================================
============
Private Function min(Num1 As Single, Num2 As Single) As Single

  If Num1 < Num2 Then
    min = Num1
  Else
    min = Num2
  End If

End Function

Public Sub PrintRegistration()
    ThisWorkbook.Sheets("Registration").Select
```

```
    ActiveWindow.SelectedSheets.PrintOut copies:=1
End Sub

Public Sub SaveWorkbook(SaveWorkbook As Workbook, Optional NewName As Variant)
Dim SaveFileName        As String

    Application.DisplayAlerts = False

    If Not IsMissing(NewName) Then
      SaveFileName = SaveWorkbook.Path & "\" & NewName
      SaveWorkbook.SaveAs FileName:=SaveFileName, FileFormat:=xlNormal, _
                          Password:="", WriteResPassword:="", ReadOnlyRecommended:=F
alse, _
                          CreateBackup:=False
    Else
'   -- only save workbook if there has been as change
      If SaveWorkbook.Saved = False Then
'       -- get full file name for workbook
        SaveFileName = SaveWorkbook.FullName
'       -- allow client to save under alternate name
        SaveFileName = Application.GetSaveAsFilename(SaveFileName, "Microsoft Excel
Workbook(*.xls), *.xls")
'       -- check to see if client wants to cancel save
        If SaveFileName = "False" Then
          Exit Sub
        Else
          SaveWorkbook.SaveAs FileName:=SaveFileName, FileFormat:=xlNormal, _
                              Password:="", WriteResPassword:="", ReadOnlyRecommende
d:=False, _
                              CreateBackup:=False
        End If
      End If
    End If

    Application.DisplayAlerts = True

End Sub

'==================================================================================
============
Public Sub SelectAllWireCenters()

    SelectListItems DialogSheets("dlgReviewSelection").ListBoxes(1), True

End Sub

'==================================================================================
============
Public Sub SelectListItem(lstAny As Object, target As String)
Dim index         As Long

  For index = 1 To lstAny.ListCount
    If lstAny.List(index) = target Then
      If TypeName(lstAny) = "Listbox" Then
        If lstAny.MultiSelect = xlNone Then
          lstAny.Selected(index) = True
        Else
          lstAny.ListIndex = index
        End If
      Else
        lstAny.ListIndex = index
```

```
      End If
      Exit For
    End If
  Next index

End Sub

'==================================================================================
============
Public Sub SelectListItems(lstAny As Listbox, flag As Boolean)
Dim index        As Long
Dim Choices()    As Boolean

    If lstAny.ListCount > 0 Then
        ReDim Choices(lstAny.ListCount)

        For index = 1 To lstAny.ListCount
            Choices(index) = flag
        Next index

        lstAny.Selected = Choices
    End If

End Sub

'==================================================================================
============
Public Sub ClearStateSelections()
Dim index                As Long
Dim List                 As Listbox
Dim Choices()            As Boolean

  On Local Error Resume Next

  Set List = ThisWorkbook.ActiveSheet.ListBoxes("lstStates")

  If List.MultiSelect = xlNone Then
    List.ListIndex = 0
  Else
    ReDim Choices(List.ListCount)
    List.Selected = Choices
  End If

End Sub

Public Sub ClearReviewStateSelection()

  ClearStateSelections
  ThisWorkbook.Sheets("review").ListBoxes("lstWireCenters").RemoveAllItems

End Sub

'==================================================================================
============
Public Sub SetPaths(Optional State As Variant)
Dim RootPath         As String
Dim RangeSheet       As Worksheet
Dim rng              As Range
Dim Cosa             As String

    Set RangeSheet = ThisWorkbook.Sheets("RangeNames")
```

```
    --reset all paths to default values
    RootPath = ThisWorkbook.Path & "\"
    RangeSheet.Range("Model_Path") = RootPath & "logic\"
    RangeSheet.Range("Output_Path") = RootPath & "results\"
    RangeSheet.Range("Input_Path") = RootPath & "inputs\"
    RangeSheet.Range("Factor_Path") = RootPath & "factors\"
    RangeSheet.Range("Model_Inf_File") = RootPath & "models.inf"

    --build working directories if State is passed
    If Not IsMissing(State) Then
      If Len(State) > 2 Then
        --assume full state name has been passed
        Set rng = RangeSheet.Range("StateList").Find(State)
        Cosa = rng.Offset(0, -1).Text
      Else
        Cosa = State
      End If
      RangeSheet.Range("Output_Path") = RootPath & "results\" & Cosa & "\"
      RangeSheet.Range("Input_Path") = RootPath & "inputs\" & Cosa & "\"
    End If

    Set RangeSheet = Nothing

End Sub

'==================================================================================
============
Public Sub ShowMessage(Optional Msg As Variant)
Dim ChangedUpdating       As Boolean
Dim wb                    As Workbook

  On Local Error Resume Next

  Set wb = ActiveWorkbook
  ThisWorkbook.Activate

  If Application.ScreenUpdating = False Then
    Application.ScreenUpdating = True
    ChangedUpdating = True
  End If

  If IsMissing(Msg) Then
    ActiveSheet.TextBoxes("txtMessage").Text = ""
  Else
    ActiveSheet.TextBoxes("txtMessage").Text = Msg
  End If

  If ChangedUpdating Then
    Application.ScreenUpdating = Not Application.ScreenUpdating
  End If

  wb.Activate
  Set wb = Nothing

End Sub

'==================================================================================
============
Public Sub ShowStatusMessage(Optional Msg As Variant)
```

```
    If IsMissing(Msg) Then
        Application.StatusBar = False
    Else
        Application.StatusBar = Msg
    End If

End Sub

'==========================================================================
============
Public Sub ShowThings()


    SetDisplay True

End Sub

'==========================================================================
============
Public Sub ShowWireCenterData(TargetWireCenter As String)
Dim index                   As Long, _
    NumberOfRows            As Long
Dim InputFile               As String
Dim tempRange               As Range, _
    outRange                As Range
Dim ws                      As Worksheet, _
    wsOut                   As Worksheet
Dim LastRow                 As Long, _
    FirstRow                As Long, _
    FirstCol                As Long, _
    LastCol                 As Long, _
    InputColumns            As Long, _
    OutLastRow              As Long, _
    OutFirstRow             As Long, _
    OutFirstCol             As Long, _
    OutLastCol              As Long, _
    OutputColumns           As Long
Dim SingleRow               As Boolean
Dim Data                    As Variant
Static ClearRanges(3)       As Range
Static SomethingReviewed    As Boolean

' --dimension array to hold ranges to be cleared
  For index = 1 To 3
    On Error Resume Next
    ClearRanges(index).ClearContents
  Next index
  On Error GoTo 0

' --get data associated with chosen wire center
  InputFile = ThisWorkbook.Sheets("RangeNames").Range("Input_Path") & CurrentModel.C
BGDataSet
  SetIndexFile InputFile
  Data = GetWireCenterData(InputFile, TargetWireCenter)

  MoveWireCenterData Data
  NumberOfRows = UBound(Data)

  SingleRow = (NumberOfRows = 1)

' -- copy formulas, write results
```

```
  Set ws = ActiveWorkbook.Worksheets("Main Logic")
  Set tempRange = ws.Range("MainCalculations")
  ws.Activate

  FirstRow = tempRange.row
  LastRow = FirstRow + NumberOfRows - 1
  FirstCol = tempRange.Column
  LastCol = FirstCol + tempRange.columns.Count
  InputColumns = ThisWorkbook.Sheets("RangeNames").Range("Max_Input_Columns")

' --copy formulas if more than one row
  If Not SingleRow Then
    ws.Range(Cells(FirstRow, tempRange.Column), Cells(LastRow, tempRange.Column + te
mpRange.columns.Count)).FillDown
  End If

' --sort data in main logic
  SortCalculations FirstRow, LastRow

  ws.Calculate

' --clear first row
  Set ClearRanges(1) = ws.Range(Cells(FirstRow, 1), Cells(LastRow, InputColumns))

' --add range to clear range array
  If Not SingleRow Then
'   --clear everything but first row
    Set ClearRanges(2) = ws.Range(Cells(FirstRow + 1, 1), Cells(LastRow, LastCol))
  End If

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

' --define boundary rows, cols for 'outputs" sheet
  OutFirstRow = outRange.row
  OutLastRow = OutFirstRow + NumberOfRows - 1
  OutFirstCol = outRange.Column
  OutLastCol = OutFirstCol + outRange.columns.Count

  If Not SingleRow Then
    wsOut.Range(Cells(OutFirstRow, OutFirstCol), Cells(OutLastRow, OutLastCol)).Fill
Down
  End If

  Set ClearRanges(3) = wsOut.Range(Cells(OutFirstRow + 1, OutFirstCol), Cells(OutLas
tRow, OutLastCol))

  wsOut.Calculate

  Set tempRange = Nothing
  Set outRange = Nothing
  Set ws = Nothing
  Set wsOut = Nothing

End Sub

' ==============================================
' Screen display routines
' ==============================================
```

```
Public Sub HideThings(ws As Worksheet)

  SetDisplay False
  MoveControls ws

End Sub

Public Sub HideTimeTextBox()

  ThisWorkbook.Sheets("Process").TextBoxes("tbTime").Visible = False

End Sub

Private Sub SetDisplay(DisplayFlag As Boolean)

  On Error Resume Next

  With ActiveWindow
    .DisplayHeadings = DisplayFlag
    .DisplayHorizontalScrollBar = DisplayFlag
    .DisplayVerticalScrollBar = DisplayFlag
  End With

End Sub

Private Sub SizeHomeCell(ws As Worksheet)
Dim Decrement       As Single

    On Error GoTo SizeHomeCell_Error

    Decrement = 50

    ws.Cells(1, 1).ColumnWidth = 254
    ws.Cells(1, 1).RowHeight = ActiveWindow.UsableHeight + Decrement
    Exit Sub

SizeHomeCell_Error:

    If Err = 1005 Then
      Decrement = Decrement - 50
      Resume
    End If

End Sub

Public Sub MoveControls(ByVal sht As Worksheet)
Dim ControlHeight         As Single, _
    ControlWidth          As Single, _
    ControlLeft           As Single, _
    ControlTop            As Single, _
    LeftPosition          As Single, _
    TopPosition           As Single, _
    MaxWidth              As Single, _
    MaxHeight             As Single
Dim LeftShift             As Long, _
    DownShift             As Long
Dim cntl                  As Object

' --set control area height
  ControlHeight = sht.GroupBoxes("Main Control").Height
  ControlLeft = sht.GroupBoxes("Main Control").Left
```

```
  ControlTop = sht.GroupBoxes("Main Control").Top
  ControlWidth = sht.GroupBoxes("Main Control").Width

  MaxWidth = ActiveWindow.UsableWidth
  MaxHeight = min(ActiveWindow.UsableHeight, Cells(1, 1).RowHeight)

  LeftPosition = (MaxWidth - ControlWidth) / 2
  TopPosition = (MaxHeight - ControlHeight) / 2

  LeftShift = LeftPosition - ControlLeft
  DownShift = TopPosition - ControlTop

  For Each cntl In sht.GroupBoxes
    cntl.Top = cntl.Top + DownShift
    cntl.Left = cntl.Left + LeftShift
  Next cntl

  For Each cntl In sht.ListBoxes
    cntl.Top = cntl.Top + DownShift
    cntl.Left = cntl.Left + LeftShift
  Next cntl

  For Each cntl In sht.TextBoxes
    cntl.Top = cntl.Top + DownShift
    cntl.Left = cntl.Left + LeftShift
  Next cntl

  For Each cntl In sht.Buttons
    cntl.Top = cntl.Top + DownShift
    cntl.Left = cntl.Left + LeftShift
  Next cntl

  For Each cntl In sht.Labels
    cntl.Top = cntl.Top + DownShift
    cntl.Left = cntl.Left + LeftShift
  Next cntl

  For Each cntl In sht.DropDowns
    cntl.Top = cntl.Top + DownShift
    cntl.Left = cntl.Left + LeftShift
  Next cntl

  For Each cntl In sht.OptionButtons
    cntl.Top = cntl.Top + DownShift
    cntl.Left = cntl.Left + LeftShift
  Next cntl

  For Each cntl In sht.CheckBoxes
    cntl.Top = cntl.Top + DownShift
    cntl.Left = cntl.Left + LeftShift
  Next cntl

  Set cntl = Nothing

End Sub

Public Sub SwitchOnTabs()

    With ActiveWindow
        .DisplayWorkbookTabs = True
        .DisplayVerticalScrollBar = True
```

```
    End With
End Sub
```

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

' --Constants associated with Primary Key
  Private Const PRIMARY_NATIONAL = 1
  Private Const PRIMARY_HOLDINGCOMPANY = 4
  Private Const PRIMARY_COMPANY = 2
  Private Const PRIMARY_STATE = 8

  Private Const SECONDARY_ONE = 1
  Private Const SECONDARY_TWO = 2

  Private Const PRIMARY = 1
  Private Const SECONDARY = 2

  Private Const REVIEW_SUMMARY = 1
  Private Const PRINT_SUMMARY = 2
  Private Const PRINT_INDIVIDUAL = 3

  Private PrimaryKey          As Long
  Private SecondaryKey        As Long
  Private LastOption          As Long
  Private SaveReports         As Boolean


' ==============================================================================
==
Public Sub ReportPrint()
Dim chk1            As CheckBox
Dim chk2            As CheckBox

  Set chk1 = ThisWorkbook.Sheets("Options").CheckBoxes("SummaryReports")
  Set chk2 = ThisWorkbook.Sheets("Options").CheckBoxes("IndividualReports")

  If CheckReportSelections() Then
    If chk1.Value = xlOn Then
      ProcessReports PRINT_SUMMARY
    End If
    If chk2.Value = xlOn Then
      ProcessReports PRINT_INDIVIDUAL
    End If
  End If

  ThisWorkbook.Activate

  Set chk1 = Nothing
  Set chk2 = Nothing

End Sub

Public Sub ReportReview()

  If CheckReportSelections() Then
    ProcessReports REVIEW_SUMMARY
    p_ReturningFromWB = True
  End If

End Sub

Public Sub MoveDataToReports(StateDescription As String, ReportType As String)
Dim sht             As Worksheet
Dim IniFile         As String
```

```
Dim dd                As DropDown

  If LoadReportWorkbook Then
    Set dd = ThisWorkbook.Sheets("Reports").DropDowns("FactorSets")
    IniFile = ThisWorkbook.Sheets("rangenames").Range("factor_path") & dd.List(dd.Li
stIndex)
    ShowStatusMessage "loading data into reports"
    CopyReportData
    CopyCapitalCostData IniFile
    CopyExpenseData IniFile
    CopyReportInformation StateDescription, ReportType, dd.List(dd.ListIndex)
'   --calculate all sheets
    Calculate
'   --use standard footer
    For Each sht In ActiveWorkbook.Worksheets
      With sht.PageSetup
        If Application.OperatingSystem <> "Windows (32-bit) 4.00" Then
          .LeftFooter = "&F.xls"
        Else
          .LeftFooter = "&F"
        End If
        .CenterFooter = "Page &P"
        .RightFooter = "&D &T"
      End With
    Next sht
    ShowStatusMessage
    Set dd = Nothing
  End If

End Sub

Public Sub PrintReports()
Dim lst                As Listbox
Dim index              As Long

  Set lst = ThisWorkbook.Sheets("Options").ListBoxes("lstReports")

  For index = 1 To lst.ListCount
    If lst.Selected(index) Then
      Sheets(lst.List(index)).PrintOut copies:=1
    End If
  Next index

  Set lst = Nothing

End Sub

Public Sub ShowAnalysisSheet()

  If ShowDetail Then
    ThisWorkbook.Sheets("Aggregate Support Calculations").Activate
  Else
    Sheets("Analysis").Activate
  End If

End Sub

Public Sub ProcessReports(Switch As Long)
Dim tbPrimary            As Textbox
Dim Files()              As String, _
    SingleFile(1)        As String
```

```
Dim IniFile              As String, _
    StateFile            As String, _
    StateName            As String
Dim Targets()            As Variant
Dim SingleTarget(1)      As Variant
Dim index                As Long
Dim FoundCell            As Range

  Application.ScreenUpdating = False

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

  SaveReports = (ThisWorkbook.Sheets("Options").CheckBoxes("SaveReports").Value = xl
On)

' --assume no aggregate support detail is going to be saved
  SaveDetail = False

  Select Case PrimaryKey
    Case PRIMARY_NATIONAL
      CheckAllStateFiles Files()
      Select Case Switch
        Case PRINT_INDIVIDUAL
          SaveDetail = (ThisWorkbook.Sheets("Options").CheckBoxes("AggregateSupportD
etail").Value = xlOn)
          For index = 1 To UBound(Files)
            StateName = FullStateName(Files(index))
            SingleFile(1) = Files(index)
            RollupFiles SingleFile
            MoveDataToReports "Total State", StateName
            PrintReports
            If SaveReports Then
              Set FoundCell = ThisWorkbook.Worksheets("RangeNames").Range("StateList
").Find(what:=StateName, lookat:=xlWhole)
              SaveReportWorkbook FoundCell.Offset(0, -1).Cells(1, 1).Text
            End If
          Next index
        Case PRINT_SUMMARY
          RollupFiles Files
          MoveDataToReports "National", SetStateString()
          PrintReports
          If SaveReports Then
            SaveReportWorkbook "National"
          End If
        Case Else
          RollupFiles Files
          MoveDataToReports "National", SetStateString()
          ShowAnalysisSheet
        End Select
    Case PRIMARY_HOLDINGCOMPANY, PRIMARY_COMPANY
'     --PrimaryKey can be used to tell which field in Record to use as key.
      Select Case SecondaryKey
        Case SECONDARY_TWO
'         --process selected states
          If Not (CheckSelectedStateFiles(Files()) > 0) Then
            MsgBox "Either no items selected from list, or no matching data found fo
r current selection"
            Exit Sub
          Else
            Select Case Switch
              Case PRINT_INDIVIDUAL
```

```
                For index = 1 To UBound(Files)
                  StateName = FullStateName(Files(index))
                  SingleFile(1) = Files(index)
                  RollupFiles SingleFile, tbPrimary.Text, PrimaryKey
                  MoveDataToReports tbPrimary.Text, StateName
                  PrintReports
                Next index
              Case PRINT_SUMMARY
                RollupFiles Files, tbPrimary.Text, PrimaryKey
                MoveDataToReports tbPrimary.Text, SetStateString()
                PrintReports
              Case Else
                RollupFiles Files, tbPrimary.Text, PrimaryKey
                MoveDataToReports tbPrimary.Text, SetStateString()
                ShowAnalysisSheet
            End Select
          End If
        Case Else
          --process all states
          CheckAllStateFiles Files()
          RollupFiles Files, tbPrimary.Text, PrimaryKey
          MoveDataToReports tbPrimary.Text, SetStateString()
          If Switch <> REVIEW_SUMMARY Then
            PrintReports
          Else
            ShowAnalysisSheet
          End If
      End Select
    Case PRIMARY_STATE
      StateFile = SetStateFile
      Select Case SecondaryKey
        Case SECONDARY_ONE
          --process selected Wire Centers
          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, 1
                  MoveDataToReports tbPrimary.Text, CStr(Targets(index))
                  PrintReports
                Next index
              Case PRINT_SUMMARY
                RollupFile StateFile, Targets, 1
                If UBound(Targets) > 1 Then
                  MoveDataToReports tbPrimary.Text, "Multiple Wire Centers [" & UBou
nd(Targets) & "]"
                Else
                  MoveDataToReports tbPrimary.Text, "[" & Targets(1) & "]"
                End If
                PrintReports
              Case Else
                RollupFile StateFile, Targets, 1
                If UBound(Targets) > 1 Then
                  MoveDataToReports tbPrimary.Text, "Multiple Wire Centers [" & UBou
nd(Targets) & "]"
                Else
                  MoveDataToReports tbPrimary.Text, "[" & Targets(1) & "]"
```