Public Sub CHIPTOP\_SHMEM\_STATUS()

On Error GoTo ErrorHandler

' Base output path

Dim basePath As String

basePath = "<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\TP_COVERAGE_SHEETS\>"

' Define sheet filters

Dim excludedSheets As Variant, includedSheets As Variant, matchedSheets As Collection

excludedSheets = Array("TP\_COVERAGE\_UPF", "TP\_COVERAGE\_SHMEM\_SDF", "TP\_COVERAGE\_CT\_SDF", "TP\_COVERAGE\_CT\_TYP\_MAX", "TP\_COVERAGE\_CT\_TYP\_MIN", \_

"TP\_COVERAGE\_SHMEM\_TYP\_MAX", "TP\_COVERAGE\_SHMEM\_TYP\_MIN", "TP\_COVERAGE\_CT\_TYP\_MAX\_MAX", \_

"TP\_COVERAGE\_CT\_TYP\_MIN\_MIN", "TP\_COVERAGE\_SHMEM\_TYP\_MAX\_MAX", "TP\_COVERAGE\_SHMEM\_TYP\_MIN\_MIN", \_

"TP\_COVERAGE\_PG0\_DELAY", "TP\_COVERAGE\_SH\_MEM\_UPF", "TP\_COVERAGE\_SH\_MEM\_PG0\_DELAY", \_

"TP\_COVERAGE\_SHMEM\_SDF\_TYP\_MIN", "TP\_COVERAGE\_SHMEM\_SDF\_TYP\_MAX", "TP\_COVERAGE\_CT\_SDF\_TYP\_MIN", \_

"TP\_COVERAGE\_CT\_SDF\_TYP\_MAX", "TP\_COVERAGE\_GLS\_SDF", "GLS\_SDF\_SANITY\_HP", "CHIP\_TOP", "MBIST", \_

"PINMUX", "apb\_i2c", "OSPI\_SS", "JPEG\_SS", "MRAM\_SS", "ZAPHOD\_SS", "ISP\_SS", "SHARED\_MEM\_SS", \_

"MRAM\_RC\_SS", "CVM\_OCVM\_SS", "CPI\_ISP\_SS", "MRAM\_SS\_OLD", "OSPI\_SS\_plan", "JPEG\_SS\_OLD", \_

"TP\_COVERAGE\_SHMEM\_PG0\_DELAY", "10Jan\_TP\_COVERAGE\_SDF\_SANITY\_HP", "TP\_COVERAGE\_SDF\_SANITY\_HP", \_

"TP\_COVERAGE\_SHMEM\_UPF", "TP\_COVERAGE\_SHMEM\_SDF")

' Get sheets ending with \_SS

Set matchedSheets = New Collection

Dim ws As Worksheet

For Each ws In ThisWorkbook.Sheets

If Right(ws.Name, 3) = "\_SS" Then

matchedSheets.Add ws.Name

End If

Next ws

ReDim includedSheets(1 To matchedSheets.Count)

Dim i As Long

For i = 1 To matchedSheets.Count

includedSheets(i) = matchedSheets(i)

Next i

' Create and setup workbooks

Dim upfWB As Workbook, pg0WB As Workbook, shMemUpfWB As Workbook, shMemPg0WB As Workbook

Set upfWB = Workbooks.Add

Set pg0WB = Workbooks.Add

Set shMemUpfWB = Workbooks.Add

Set shMemPg0WB = Workbooks.Add

' Setup headers for all workbooks

With upfWB.Sheets(1)

.Cells(1, 1).Value = "A"

.Cells(1, 2).Value = ""

.Cells(1, 10).Value = "Owner"

.Cells(1, 6).Value = "SV Test Name"

.Cells(1, 7).Value = "C Test Name"

.Cells(1, 8).Value = "Test ID"

.Cells(1, 3).Value = "Current Status"

.Cells(1, 4).Value = "Previous Status"

.Cells(1, 5).Value = "CPU"

End With

With pg0WB.Sheets(1)

.Cells(1, 1).Value = "A"

.Cells(1, 2).Value = ""

.Cells(1, 9).Value = "Owner"

.Cells(1, 5).Value = "SV Test Name"

.Cells(1, 6).Value = "C Test Name"

.Cells(1, 7).Value = "Test ID"

.Cells(1, 3).Value = "Current Status"

.Cells(1, 4).Value = "CPU"

End With

With shMemUpfWB.Sheets(1)

.Cells(1, 1).Value = "A"

.Cells(1, 2).Value = ""

.Cells(1, 9).Value = "Owner"

.Cells(1, 6).Value = "SV Test Name"

.Cells(1, 7).Value = "Test ID"

.Cells(1, 3).Value = "Current Status"

.Cells(1, 4).Value = "Previous Status"

.Cells(1, 5).Value = "CPU"

End With

With shMemPg0WB.Sheets(1)

.Cells(1, 1).Value = "A"

.Cells(1, 2).Value = ""

.Cells(1, 8).Value = "Owner"

.Cells(1, 5).Value = "SV Test Name"

.Cells(1, 6).Value = "Test ID"

.Cells(1, 3).Value = "Current Status"

.Cells(1, 4).Value = "CPU"

End With

' Initialize row counters

Dim upfRow As Long, pg0Row As Long, shMemUpfRow As Long, shMemPg0Row As Long

upfRow = 2

pg0Row = 2

shMemUpfRow = 2

shMemPg0Row = 2

' Process each worksheet

For Each ws In ThisWorkbook.Worksheets

' Check if sheet should be processed

Dim isSharedMemSheet As Boolean, isValidSheet As Boolean, k As Long

' Check if sheet is in includedSheets

isSharedMemSheet = False

For k = LBound(includedSheets) To UBound(includedSheets)

If ws.Name = CStr(includedSheets(k)) Then

isSharedMemSheet = True

Exit For

End If

Next k

' Check if sheet is not in excludedSheets

isValidSheet = True

For k = LBound(excludedSheets) To UBound(excludedSheets)

If ws.Name = CStr(excludedSheets(k)) Then

isValidSheet = False

Exit For

End If

Next k

' Find header row and Test ID column

Dim headerRow As Long, testIDCol As Long, j As Long, lastRow As Long

headerRow = 0

testIDCol = 0

lastRow = ws.Cells(ws.Rows.Count, 1).End(xlUp).row

For i = 1 To lastRow

For j = 1 To ws.Cells(i, ws.Columns.Count).End(xlToLeft).Column

If Trim(CStr(ws.Cells(i, j).Value)) = "Test ID" Then

headerRow = i

testIDCol = j

Exit For

End If

Next j

If headerRow > 0 Then Exit For

Next i

If headerRow > 0 Then

' Find column indices

Dim ownerCol As Long, svTestCol As Long, cTestCol As Long

Dim currentUPFCol As Long, cpuCol As Long

ownerCol = 0: svTestCol = 0: cTestCol = 0

currentUPFCol = 0: cpuCol = 0

For j = 1 To ws.Cells(headerRow, ws.Columns.Count).End(xlToLeft).Column

Select Case Trim(CStr(ws.Cells(headerRow, j).Value))

Case "Owner": ownerCol = j

Case "SV Test Name": svTestCol = j

Case "C Test Name": cTestCol = j

Case "Current UPF": currentUPFCol = j

Case "CPU": cpuCol = j

End Select

Next j

' Copy data

For i = headerRow + 1 To lastRow

If Not IsEmpty(ws.Cells(i, testIDCol)) Then

If isSharedMemSheet Then

' Copy to Shared Memory workbooks

With shMemUpfWB.Sheets(1)

If ownerCol > 0 Then .Cells(shMemUpfRow, 9).Value = CStr(ws.Cells(i, ownerCol).Value)

If svTestCol > 0 Then .Cells(shMemUpfRow, 6).Value = CStr(ws.Cells(i, svTestCol).Value)

.Cells(shMemUpfRow, 7).Value = CStr(ws.Cells(i, testIDCol).Value)

If currentUPFCol > 0 Then .Cells(shMemUpfRow, 4).Value = CStr(ws.Cells(i, currentUPFCol).Value)

If cpuCol > 0 Then .Cells(shMemUpfRow, 5).Value = CStr(ws.Cells(i, cpuCol).Value)

End With

With shMemPg0WB.Sheets(1)

If ownerCol > 0 Then .Cells(shMemPg0Row, 8).Value = CStr(ws.Cells(i, ownerCol).Value)

If svTestCol > 0 Then .Cells(shMemPg0Row, 5).Value = CStr(ws.Cells(i, svTestCol).Value)

.Cells(shMemPg0Row, 6).Value = CStr(ws.Cells(i, testIDCol).Value)

If cpuCol > 0 Then .Cells(shMemPg0Row, 4).Value = CStr(ws.Cells(i, cpuCol).Value)

End With

shMemUpfRow = shMemUpfRow + 1

shMemPg0Row = shMemPg0Row + 1

ElseIf isValidSheet Then

' Copy to main workbooks

With upfWB.Sheets(1)

If ownerCol > 0 Then .Cells(upfRow, 10).Value = CStr(ws.Cells(i, ownerCol).Value)

If svTestCol > 0 Then .Cells(upfRow, 6).Value = CStr(ws.Cells(i, svTestCol).Value)

If cTestCol > 0 Then .Cells(upfRow, 7).Value = CStr(ws.Cells(i, cTestCol).Value)

.Cells(upfRow, 8).Value = CStr(ws.Cells(i, testIDCol).Value)

If currentUPFCol > 0 Then .Cells(upfRow, 4).Value = CStr(ws.Cells(i, currentUPFCol).Value)

If cpuCol > 0 Then .Cells(upfRow, 5).Value = CStr(ws.Cells(i, cpuCol).Value)

End With

With pg0WB.Sheets(1)

If ownerCol > 0 Then .Cells(pg0Row, 9).Value = CStr(ws.Cells(i, ownerCol).Value)

If svTestCol > 0 Then .Cells(pg0Row, 5).Value = CStr(ws.Cells(i, svTestCol).Value)

If cTestCol > 0 Then .Cells(pg0Row, 6).Value = CStr(ws.Cells(i, cTestCol).Value)

.Cells(pg0Row, 7).Value = CStr(ws.Cells(i, testIDCol).Value)

If cpuCol > 0 Then .Cells(pg0Row, 4).Value = CStr(ws.Cells(i, cpuCol).Value)

End With

upfRow = upfRow + 1

pg0Row = pg0Row + 1

End If

End If

Next i

End If

Next ws

' Process SDF files

Dim exportConfigs As Variant

exportConfigs = Array( \_

Array("TP\_COVERAGE\_CT\_TYP\_MAX.csv", excludedSheets, True, 9, 7, False, 10, 4), \_

Array("TP\_COVERAGE\_CT\_TYP\_MIN.csv", excludedSheets, True, 9, 7, False, 12, 6), \_

Array("TP\_COVERAGE\_CT\_TYP\_MAX\_MAX.csv", excludedSheets, True, 9, 7, False, 14, 8), \_

Array("TP\_COVERAGE\_CT\_TYP\_MIN\_MIN.csv", excludedSheets, True, 9, 7, False, 16, 10), \_

Array("TP\_COVERAGE\_SHMEM\_TYP\_MAX.csv", includedSheets, False, 8, 6, True, 9, 4), \_

Array("TP\_COVERAGE\_SHMEM\_TYP\_MIN.csv", includedSheets, False, 8, 6, True, 11, 6), \_

Array("TP\_COVERAGE\_SHMEM\_TYP\_MAX\_MAX.csv", includedSheets, False, 8, 6, True, 13, 8), \_

Array("TP\_COVERAGE\_SHMEM\_TYP\_MIN\_MIN.csv", includedSheets, False, 8, 6, True, 15, 10) \_

)

' Process each SDF configuration

Dim config As Variant

For Each config In exportConfigs

Dim exportWB As Workbook

Set exportWB = Workbooks.Add

' Setup headers

With exportWB.Sheets(1)

.Cells(1, 1).Value = "A"

.Cells(1, 2).Value = ""

.Cells(1, 3).Value = ""

Select Case config(7) ' targetCol

Case 4: .Cells(1, 4).Value = "TYP\_MAX Data"

Case 6: .Cells(1, 6).Value = "TYP\_MIN Data"

Case 8: .Cells(1, 8).Value = "MAX\_MAX Data"

Case 10: .Cells(1, 10).Value = "MIN\_MIN Data"

End Select

If config(2) Then ' isTYP\_MAX

.Cells(1, 11).Value = "SV Test Name"

.Cells(1, 12).Value = "C Test Name"

.Cells(1, 13).Value = "Test ID"

.Cells(1, 15).Value = "Owner"

Else

.Cells(1, 11).Value = "SV Test Name"

.Cells(1, 12).Value = "Test ID"

.Cells(1, 14).Value = "Owner"

End If

End With

' Process worksheets

Dim exportRow As Long: exportRow = 2

For Each ws In ThisWorkbook.Worksheets

' Check if sheet should be processed

Dim shouldProcess As Boolean

shouldProcess = False

If config(5) Then ' isInclusiveFilter

For k = LBound(config(1)) To UBound(config(1))

If ws.Name = CStr(config(1)(k)) Then

shouldProcess = True

Exit For

End If

Next k

Else

shouldProcess = True

For k = LBound(config(1)) To UBound(config(1))

If ws.Name = CStr(config(1)(k)) Then

shouldProcess = False

Exit For

End If

Next k

End If

If shouldProcess Then

' Find header row and Test ID column

headerRow = 0: testIDCol = 0

lastRow = ws.Cells(ws.Rows.Count, 1).End(xlUp).row

For i = 1 To lastRow

For j = 1 To ws.Cells(i, ws.Columns.Count).End(xlToLeft).Column

If Trim(CStr(ws.Cells(i, j).Value)) = "Test ID" Then

headerRow = i

testIDCol = j

Exit For

End If

Next j

If headerRow > 0 Then Exit For

Next i

If headerRow > 0 Then

' Find necessary columns

svTestCol = 0: cTestCol = 0: ownerCol = 0

For j = 1 To ws.Cells(headerRow, ws.Columns.Count).End(xlToLeft).Column

Select Case Trim(CStr(ws.Cells(headerRow, j).Value))

Case "SV Test Name": svTestCol = j

Case "C Test Name": cTestCol = j

Case "Owner": ownerCol = j

End Select

Next j

' Copy matching data

For i = headerRow + 1 To lastRow

If Not IsEmpty(ws.Cells(i, testIDCol)) And \_

Not IsEmpty(ws.Cells(i, ownerCol)) And \_

LCase(Trim(CStr(ws.Cells(i, config(3)).Value))) = "yes" Then

' Verify Test ID and Owner contain actual values

Dim testID As String, owner As String

testID = Trim(CStr(ws.Cells(i, testIDCol).Value))

owner = Trim(CStr(ws.Cells(i, ownerCol).Value))

If testID <> "" And owner <> "" Then

With exportWB.Sheets(1)

' Add extracted data in the correct target column

If config(6) > 0 And config(7) > 0 Then

.Cells(exportRow, config(7)).Value = CStr(ws.Cells(i, config(6)).Value)

End If

' Dynamic columns based on type

If config(2) Then ' isTYP\_MAX

If svTestCol > 0 Then .Cells(exportRow, 11).Value = CStr(ws.Cells(i, svTestCol).Value)

If cTestCol > 0 Then .Cells(exportRow, 12).Value = CStr(ws.Cells(i, cTestCol).Value)

.Cells(exportRow, 13).Value = testID

.Cells(exportRow, 15).Value = owner

Else

If svTestCol > 0 Then .Cells(exportRow, 11).Value = CStr(ws.Cells(i, svTestCol).Value)

.Cells(exportRow, 12).Value = testID

.Cells(exportRow, 14).Value = owner

End If

End With

exportRow = exportRow + 1

End If

End If

Next i

End If

End If

Next ws

' Save and close SDF workbook

exportWB.SaveAs Filename:=basePath & config(0), FileFormat:=xlCSV, CreateBackup:=False

exportWB.Close SaveChanges:=False

Next config

' Save and close main workbooks

upfWB.SaveAs Filename:=basePath & "TP\_COVERAGE\_UPF.csv", FileFormat:=xlCSV, CreateBackup:=False

pg0WB.SaveAs Filename:=basePath & "TP\_COVERAGE\_PG0\_DELAY.csv", FileFormat:=xlCSV, CreateBackup:=False

shMemUpfWB.SaveAs Filename:=basePath & "TP\_COVERAGE\_SH\_MEM\_UPF.csv", FileFormat:=xlCSV, CreateBackup:=False

shMemPg0WB.SaveAs Filename:=basePath & "TP\_COVERAGE\_SH\_MEM\_PG0\_DELAY.csv", FileFormat:=xlCSV, CreateBackup:=False

upfWB.Close SaveChanges:=False

pg0WB.Close SaveChanges:=False

shMemUpfWB.Close SaveChanges:=False

shMemPg0WB.Close SaveChanges:=False

Call UpdateTPCoverageAll

Exit Sub

ErrorHandler:

MsgBox "Error: " & Err.Description, vbCritical

End Sub

Private Sub UpdateTPCoverageAll()

' Constants for configuration

Const CONFIG\_UPF = "UPF,TP\_COVERAGE\_UPF,Regression,TP\_COVERAGE\_UPF,E,H,C,I"

Const CONFIG\_PG0 = "PG0,TP\_COVERAGE\_PG0\_DELAY,Regression\_PG0,TP\_COVERAGE\_PG0\_DELAY,D,G,C,H"

Const CONFIG\_SH\_MEM\_UPF = "SH\_MEM\_UPF,TP\_COVERAGE\_SHMEM\_UPF,Regression\_SH\_MEM,TP\_COVERAGE\_SH\_MEM\_UPF,E,G,C,H"

Const CONFIG\_SH\_MEM\_PG0 = "SH\_MEM\_PG0,TP\_COVERAGE\_SHMEM\_PG0\_DELAY,Regression\_SH\_MEM\_PG0,TP\_COVERAGE\_SH\_MEM\_PG0\_DELAY,D,F,C,G"

' Get user choice

Dim userChoice As String, ws As Worksheet

userChoice = InputBox("1. UPF" & vbNewLine & "2. PG0" & vbNewLine & "3. SH\_MEM\_UPF" & vbNewLine & \_

"4. SH\_MEM\_PG0" & vbNewLine & "5. TP\_COVERAGE\_CT\_SDF" & vbNewLine & "6. TP\_COVERAGE\_SHMEM\_SDF", \_

"Select Configuration")

If userChoice = "" Or Not IsNumeric(userChoice) Then Exit Sub

Dim choiceNum As Integer: choiceNum = CInt(userChoice)

If choiceNum < 1 Or choiceNum > 6 Then Exit Sub

' Variable to track the last row of previous data

Dim lastPreviousRow As Long

If choiceNum <= 4 Then

' Set configuration parameters based on choice

Dim configParams() As String

Select Case choiceNum

Case 1: configParams = Split(CONFIG\_UPF, ",")

Case 2: configParams = Split(CONFIG\_PG0, ",")

Case 3: configParams = Split(CONFIG\_SH\_MEM\_UPF, ",")

Case 4: configParams = Split(CONFIG\_SH\_MEM\_PG0, ",")

End Select

' Setup worksheet

On Error Resume Next

Set ws = ThisWorkbook.Sheets(configParams(1))

On Error GoTo 0

Dim startRow As Long

If ws Is Nothing Then

Set ws = ThisWorkbook.Sheets.Add(Before:=ThisWorkbook.Sheets(1))

ws.Name = configParams(1)

startRow = 1

lastPreviousRow = 0

Else

' 1. FIRST PROCESS ROWS: Compact data by removing rows with blank values in key column

Dim r As Long, writeRow As Long

writeRow = 1

' Get initial estimate of last row (will be refined later)

lastPreviousRow = ws.Cells(ws.Rows.Count, configParams(5)).End(xlUp).row

If lastPreviousRow > 0 Then

For r = 1 To lastPreviousRow

If Not IsEmpty(ws.Cells(r, configParams(7)).Value) And ws.Cells(r, configParams(7)).Value <> "" Then

If r <> writeRow Then

ws.Range("C" & r & ":L" & r).Copy ws.Range("C" & writeRow)

End If

writeRow = writeRow + 1

End If

Next r

If writeRow <= lastPreviousRow Then

ws.Range("C" & writeRow & ":L" & lastPreviousRow).ClearContents

End If

End If

' 2. FIND COLUMN WITH MOST DATA: Check columns A through AD for the one with data in furthest row

Dim lastRowByCol As Long, highestLastRow As Long, colIndex As Integer

highestLastRow = 0

For colIndex = 1 To 30 ' Column A to AD

lastRowByCol = ws.Cells(ws.Rows.Count, colIndex).End(xlUp).row

If lastRowByCol > highestLastRow Then

highestLastRow = lastRowByCol + 1

End If

Next colIndex

lastPreviousRow = highestLastRow

' 3. DELETE ROWS: Remove rows below the last data row

If lastPreviousRow < ws.Rows.Count Then

ws.Rows(lastPreviousRow & ":" & ws.Rows.Count).Delete

End If

' 4. HIDE DATA: Hide the processed rows

ws.Rows("1:" & (lastPreviousRow)).Hidden = True

' 5. ADD NEW DATA: Start adding after the last row with data

startRow = lastPreviousRow

ws.Move Before:=ThisWorkbook.Sheets(1)

End If

' File paths

Dim regressionFile As String: regressionFile = "<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\>" & configParams(2) & ".csv"

Dim tpTestFile As String: tpTestFile = "<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\TP_COVERAGE_SHEETS\>" & configParams(3) & ".csv"

' Open and process workbooks

Dim regressionData As Workbook, TPTestData As Workbook

Set regressionData = Workbooks.Open(regressionFile)

Set TPTestData = Workbooks.Open(tpTestFile)

If regressionData Is Nothing Or TPTestData Is Nothing Then

MsgBox "Error opening CSV files. Please check file paths.", vbCritical

Exit Sub

End If

' Copy TP Test data

With TPTestData.Sheets(1)

Dim lastRow As Long: lastRow = .Cells(.Rows.Count, configParams(5)).End(xlUp).row

' Always copy headers for each new section

.Rows(1).Copy Destination:=ws.Range("A" & startRow)

startRow = startRow + 1

' Copy data

.Range("A2:M" & lastRow).Copy Destination:=ws.Range("A" & startRow)

End With

' Match and update data

Dim regLastRow As Long: regLastRow = regressionData.Sheets(1).Cells(regressionData.Sheets(1).Rows.Count, "A").End(xlUp).row

Dim tpLastRow As Long: tpLastRow = ws.Cells(ws.Rows.Count, configParams(5)).End(xlUp).row

Dim tpRow As Long, regRow As Long, testID As String

For tpRow = startRow To tpLastRow

testID = ws.Cells(tpRow, configParams(5)).Value

For regRow = 2 To regLastRow

If regressionData.Sheets(1).Cells(regRow, "E").Value = testID Then

ws.Cells(tpRow, configParams(6)).Value = regressionData.Sheets(1).Cells(regRow, "C").Value

ws.Cells(tpRow, configParams(4)).Value = regressionData.Sheets(1).Cells(regRow, "D").Value

Exit For

End If

Next regRow

Next tpRow

regressionData.Close False: TPTestData.Close False

ApplyConditionalFormatting ws, configParams(6) & ":" & configParams(4), False

Else

' Handle SDF options

Dim sheetName As String, configs As Variant, baseFiles As Variant, testIdColumn As String, tpMissingColumn As String

If choiceNum = 5 Then

sheetName = "TP\_COVERAGE\_CT\_SDF"

configs = Array("CT\_TYP\_MAX", "CT\_TYP\_MIN", "CT\_TYP\_MAX\_MAX", "CT\_TYP\_MIN\_MIN")

testIdColumn = "M"

tpMissingColumn = "N"

Else

sheetName = "TP\_COVERAGE\_SHMEM\_SDF"

configs = Array("SHMEM\_TYP\_MAX", "SHMEM\_TYP\_MIN", "SHMEM\_TYP\_MAX\_MAX", "SHMEM\_TYP\_MIN\_MIN")

testIdColumn = "L"

tpMissingColumn = "M"

End If

baseFiles = Array()

ReDim baseFiles(3)

Dim i As Integer

For i = 0 To 3

baseFiles(i) = "<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\TP_COVERAGE_SHEETS\TP_COVERAGE_>" & configs(i) & ".csv"

Next i

' Setup worksheet

On Error Resume Next

Set ws = ThisWorkbook.Sheets(sheetName)

If ws Is Nothing Then

Set ws = ThisWorkbook.Sheets.Add(Before:=ThisWorkbook.Sheets(1))

ws.Name = sheetName

startRow = 1

lastPreviousRow = 0

Else

' Process existing data

Dim rSDF As Long, writeRowSDF As Long

writeRowSDF = 1

' Get last row of existing data

lastPreviousRow = ws.Cells(ws.Rows.Count, "L").End(xlUp).row

If lastPreviousRow > 0 Then

' Compact existing data

For rSDF = 1 To lastPreviousRow

If Not IsEmpty(ws.Cells(rSDF, tpMissingColumn).Value) And ws.Cells(rSDF, tpMissingColumn).Value <> "" Then

If rSDF <> writeRowSDF Then

ws.Range("C" & rSDF & ":P" & rSDF).Copy ws.Range("C" & writeRowSDF)

End If

writeRowSDF = writeRowSDF + 1

End If

Next rSDF

' Clear any remaining data

If writeRowSDF <= lastPreviousRow Then

ws.Range("C" & writeRowSDF & ":P" & lastPreviousRow).ClearContents

End If

End If

' Find highest row with data across all columns

Dim lastRowBySDF As Long, highestLastRowSDF As Long, colSDF As Integer

highestLastRowSDF = 0

For colSDF = 1 To 30

lastRowBySDF = ws.Cells(ws.Rows.Count, colSDF).End(xlUp).row

If lastRowBySDF > highestLastRowSDF Then

highestLastRowSDF = lastRowBySDF + 1

End If

Next colSDF

lastPreviousRow = highestLastRowSDF

' Delete excess rows

If lastPreviousRow < ws.Rows.Count Then

ws.Rows(lastPreviousRow & ":" & ws.Rows.Count).Delete

End If

' Hide processed rows

ws.Rows("1:" & (lastPreviousRow)).Hidden = True

' Set start row for new data

startRow = lastPreviousRow + 1

ws.Move Before:=ThisWorkbook.Sheets(1)

End If

' Process each configuration

For i = 0 To 3

Dim baseData As Workbook

Set baseData = Workbooks.Open(baseFiles(i))

With baseData.Sheets(1)

lastRow = .Cells(.Rows.Count, "K").End(xlUp).row

If i = 0 Then

' Add headers for first configuration

.Range("A1:Q1").Copy Destination:=ws.Range("A" & startRow)

' Set custom headers based on configuration type

With ws

If choiceNum = 5 Then ' CT\_SDF

.Cells(startRow, 3).Value = "Current CT\_TYP\_MAX"

.Cells(startRow, 4).Value = "Previous CT\_TYP\_MAX"

.Cells(startRow, 5).Value = "Current CT\_TYP\_MIN"

.Cells(startRow, 6).Value = "Previous CT\_TYP\_MIN"

.Cells(startRow, 7).Value = "Current CT\_TYP\_MAX\_MAX"

.Cells(startRow, 8).Value = "Previous CT\_TYP\_MAX\_MAX"

.Cells(startRow, 9).Value = "Current CT\_TYP\_MIN\_MIN"

.Cells(startRow, 10).Value = "Previous CT\_TYP\_MIN\_MIN"

Else ' SHMEM\_SDF

.Cells(startRow, 3).Value = "Current SHMEM\_TYP\_MAX"

.Cells(startRow, 4).Value = "Previous SHMEM\_TYP\_MAX"

.Cells(startRow, 5).Value = "Current SHMEM\_TYP\_MIN"

.Cells(startRow, 6).Value = "Previous SHMEM\_TYP\_MIN"

.Cells(startRow, 7).Value = "Current SHMEM\_TYP\_MAX\_MAX"

.Cells(startRow, 8).Value = "Previous SHMEM\_TYP\_MAX\_MAX"

.Cells(startRow, 9).Value = "Current SHMEM\_TYP\_MIN\_MIN"

.Cells(startRow, 10).Value = "Previous SHMEM\_TYP\_MIN\_MIN"

End If

End With

' Copy data

.Range("A2:Q" & lastRow).Copy Destination:=ws.Range("A" & (startRow + 1))

Else

' Copy specific columns based on configuration

Select Case i

Case 1: .Range("F2:F" & lastRow).Copy Destination:=ws.Range("F" & (startRow + 1))

Case 2: .Range("H2:H" & lastRow).Copy Destination:=ws.Range("H" & (startRow + 1))

Case 3: .Range("J2:J" & lastRow).Copy Destination:=ws.Range("J" & (startRow + 1))

End Select

End If

End With

baseData.Close False

' Process regression data

regressionFile = "<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_>" & configs(i) & ".csv"

Set regressionData = Workbooks.Open(regressionFile)

regLastRow = regressionData.Sheets(1).Cells(regressionData.Sheets(1).Rows.Count, "A").End(xlUp).row

' Update regression data

For tpRow = startRow + 1 To startRow + lastRow - 1

testID = ws.Cells(tpRow, testIdColumn).Value

For regRow = 2 To regLastRow

If regressionData.Sheets(1).Cells(regRow, "E").Value = testID Then

ws.Cells(tpRow, 3 + (i \* 2)).Value = regressionData.Sheets(1).Cells(regRow, "C").Value

Exit For

End If

Next regRow

Next tpRow

regressionData.Close False

Next i

ApplyConditionalFormatting ws, "C,D,E,F,G,H,I,J", True

End If

ThisWorkbook.Save

Call STATUS

Exit Sub

End Sub

Private Function GetLastDataRow(ws As Worksheet) As Long

Dim lastRow As Long

Dim col As Long

Dim maxRow As Long

maxRow = 0

' Check all relevant columns (1 to 30) for the last row with data

For col = 1 To 30

lastRow = ws.Cells(ws.Rows.Count, col).End(xlUp).row

If lastRow > maxRow Then

maxRow = lastRow

End If

Next col

GetLastDataRow = maxRow

End Function

Private Function ApplyConditionalFormatting(ws As Worksheet, targetColumns As String, Optional isSDF As Boolean = False)

ws.Cells.FormatConditions.Delete

Dim formats As Variant

formats = Array( \_

Array("passed", RGB(198, 239, 206), RGB(0, 97, 0)), \_

Array("failed", RGB(255, 204, 204), RGB(192, 0, 0)), \_

Array("running", RGB(255, 217, 102), RGB(68, 114, 196)), \_

Array("stopped", RGB(231, 230, 230), RGB(38, 38, 38)), \_

Array("TBD", RGB(255, 242, 204), RGB(0, 0, 0)) \_

)

If isSDF Then

ws.Range("D:J").HorizontalAlignment = xlCenter

ws.Range("D:J").VerticalAlignment = xlTop

Dim colArray As Variant: colArray = Split(targetColumns, ",")

Dim col As Variant, fmt As Variant

For Each col In colArray

For Each fmt In formats

With ws.Range(col & ":" & col).FormatConditions.Add(Type:=xlTextString, String:=fmt(0), TextOperator:=xlContains)

.Interior.Color = fmt(1)

.Font.Color = fmt(2)

.Font.Bold = False

End With

Next fmt

Next col

Else

With ws.Range(targetColumns)

.HorizontalAlignment = xlCenter

.VerticalAlignment = xlTop

End With

For Each fmt In formats

With ws.Range(targetColumns).FormatConditions.Add(Type:=xlTextString, String:=fmt(0), TextOperator:=xlContains)

.Interior.Color = fmt(1)

.Font.Color = fmt(2)

.Font.Bold = False

End With

Next fmt

End If

End Function

Private Sub STATUS()

' Constants

Const SHEET\_CHIP\_TOP\_UPF As String = "TP\_COVERAGE\_UPF"

Const SHEET\_CHIP\_TOP\_PG0\_DELAY As String = "TP\_COVERAGE\_PG0\_DELAY"

Const SHEET\_SHMEM\_UPF As String = "TP\_COVERAGE\_SHMEM\_UPF"

Const SHEET\_SHMEM\_PG0\_DELAY As String = "TP\_COVERAGE\_SHMEM\_PG0\_DELAY"

Const SHEET\_CT\_SDF As String = "TP\_COVERAGE\_CT\_SDF"

Const SHEET\_SHMEM\_SDF As String = "TP\_COVERAGE\_SHMEM\_SDF"

' Global Excluded and Included Sheets (defined once)

Dim excludedSheets As Variant

Dim includedSheets As Variant

' Define excluded and included sheets once

excludedSheets = Array("TP\_COVERAGE\_UPF", "TP\_COVERAGE\_SHMEM\_SDF", "TP\_COVERAGE\_CT\_SDF", "TP\_COVERAGE\_CT\_TYP\_MAX", "TP\_COVERAGE\_CT\_TYP\_MIN", \_

"TP\_COVERAGE\_SHMEM\_TYP\_MAX", "TP\_COVERAGE\_SHMEM\_TYP\_MIN", "TP\_COVERAGE\_CT\_TYP\_MAX\_MAX", \_

"TP\_COVERAGE\_CT\_TYP\_MIN\_MIN", "TP\_COVERAGE\_SHMEM\_TYP\_MAX\_MAX", "TP\_COVERAGE\_SHMEM\_TYP\_MIN\_MIN", \_

"TP\_COVERAGE\_PG0\_DELAY", "TP\_COVERAGE\_SH\_MEM\_UPF", "TP\_COVERAGE\_SH\_MEM\_PG0\_DELAY", \_

"TP\_COVERAGE\_SHMEM\_SDF\_TYP\_MIN", "TP\_COVERAGE\_SHMEM\_SDF\_TYP\_MAX", "TP\_COVERAGE\_CT\_SDF\_TYP\_MIN", \_

"TP\_COVERAGE\_CT\_SDF\_TYP\_MAX", "TP\_COVERAGE\_GLS\_SDF", "GLS\_SDF\_SANITY\_HP", "CHIP\_TOP", "MBIST", \_

"PINMUX", "apb\_i2c", "OSPI\_SS", "JPEG\_SS", "MRAM\_SS", "ZAPHOD\_SS", "ISP\_SS", "SHARED\_MEM\_SS", \_

"MRAM\_RC\_SS", "CVM\_OCVM\_SS", "CPI\_ISP\_SS", "MRAM\_SS\_OLD", "OSPI\_SS\_plan", "JPEG\_SS\_OLD")

includedSheets = GetSheetsEndingWith("\_SS")

' Variables

Dim userChoice As String

Dim ws As Worksheet

Dim flag As Integer

Dim i As Long, j As Long

Dim lastRowSheet As Long

Dim testplanTotal As Long

Dim regressionTotal As Long

Dim tpTestsNotInRegression As Long

Dim sheetName As String

Dim makeChanges As Boolean

Dim startColumn As Integer

Dim lastHeaderColumn As Integer

Dim notFoundColumn As Integer

Dim checkColumn As Integer

Dim missingInTP As Integer

Dim CSV\_FILE\_PATH As String

Dim counter As Integer

Dim lastRowN As Integer

Dim lastRowM As Integer

Dim regressionMissingTests As Integer

Dim SVtestNameCol As Integer

' User input for macro type

userChoice = InputBox("Enter the macro to execute:" & vbCrLf & \_

"1. CHIP\_TOP\_UPF" & vbCrLf & \_

"2. CHIP\_TOP\_PG0\_DELAY" & vbCrLf & \_

"3. SHMEM\_UPF" & vbCrLf & \_

"4. SHMEM\_PG0\_DELAY" & vbCrLf & \_

"5. CT\_SDF" & vbCrLf & \_

"6. SHMEM\_SDF", "Choose Macro")

Select Case userChoice

Case "1" ' CHIP\_TOP UPF

sheetName = SHEET\_CHIP\_TOP\_UPF

makeChanges = True

CSV\_FILE\_PATH = "<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression.csv>"

lastHeaderColumn = 27

notFoundColumn = 9

checkColumn = 3

regressionMissingTests = 15

duplicateTestinRegression = 20

duplicateTestinTp = 24

SVtestNameCol = 6

Case "2" ' CHIP\_TOP PG0 DELAY

sheetName = SHEET\_CHIP\_TOP\_PG0\_DELAY

makeChanges = False

CSV\_FILE\_PATH = "<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_PG0.csv>"

lastHeaderColumn = 26

notFoundColumn = 8

checkColumn = 3

regressionMissingTests = 14

duplicateTestinRegression = 19

duplicateTestinTp = 23

SVtestNameCol = 5

Case "3" ' SHMEM UPF

sheetName = SHEET\_SHMEM\_UPF

makeChanges = True

CSV\_FILE\_PATH = "<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_SH_MEM.csv>"

lastHeaderColumn = 26

notFoundColumn = 8

checkColumn = 3

regressionMissingTests = 14

duplicateTestinRegression = 19

duplicateTestinTp = 23

SVtestNameCol = 6

Case "4" ' SHMEM PG0 DELAY

sheetName = SHEET\_SHMEM\_PG0\_DELAY

makeChanges = False

CSV\_FILE\_PATH = "<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_SH_MEM_PG0.csv>"

lastHeaderColumn = 25

notFoundColumn = 7

checkColumn = 3

regressionMissingTests = 13

duplicateTestinRegression = 18

duplicateTestinTp = 22

SVtestNameCol = 5

Case "5" ' CT\_SDF

sheetName = SHEET\_CT\_SDF

Call SDF\_STATUS("1") ' Call SDF\_STATUS with CT\_SDF option

Exit Sub

Case "6" ' SHMEM\_SDF

sheetName = SHEET\_SHMEM\_SDF

Call SDF\_STATUS("2") ' Call SDF\_STATUS with SHMEM\_SDF option

Exit Sub

Case Else

MsgBox "Invalid choice! Please enter a number between 1 and 6.", vbExclamation

Exit Sub

End Select

' Get worksheet

On Error Resume Next

Set ws = Worksheets(sheetName)

On Error GoTo 0

If ws Is Nothing Then

MsgBox "Worksheet '" & sheetName & "' not found!", vbCritical

Exit Sub

End If

statsStartRow = ws.UsedRange.row

While ws.Rows(statsStartRow).Hidden

statsStartRow = statsStartRow + 1

Wend

' Calculate totals using visible rows only

lastRowSheet = ws.Cells(ws.Rows.Count, Choose(Val(userChoice), 8, 7, 7, 6)).End(xlUp).row

While lastRowSheet > 1 And ws.Rows(lastRowSheet).Hidden

lastRowSheet = lastRowSheet - 1

Wend

testplanTotal = 0

For i = statsStartRow To lastRowSheet

If Not ws.Rows(i).Hidden Then

testplanTotal = testplanTotal + 1

End If

Next i

testplanTotal = testplanTotal - 1

' Open CSV to get regression total

Dim csvWorkbook As Workbook

On Error Resume Next

Set csvWorkbook = Workbooks.Open(CSV\_FILE\_PATH)

On Error GoTo 0

If csvWorkbook Is Nothing Then

MsgBox "Failed to open the CSV file!", vbCritical

Exit Sub

End If

' Get total rows from CSV (minus header row)

regressionTotal = csvWorkbook.Sheets(1).Cells(csvWorkbook.Sheets(1).Rows.Count, "A").End(xlUp).row - 1

csvWorkbook.Close SaveChanges:=False

' Process rows

tpTestsNotInRegression = 0

Dim visibleRowCount As Long

visibleRowCount = 1

For i = statsStartRow + 1 To lastRowSheet

' Skip hidden rows

If ws.Rows(i).Hidden Then

GoTo nextRow

End If

' Skip if already marked as "Test ID not found"

If Trim(ws.Cells(i, checkColumn).Value) = "" Then

ws.Cells(i, notFoundColumn).Value = "Test ID not found"

tpTestsNotInRegression = tpTestsNotInRegression + 1

GoTo nextRow

End If

' Process based on user choice

Select Case userChoice

Case "1"

flag = UPF\_STATUS(ws, i, excludedSheets, makeChanges)

Case "2"

flag = PG0\_DELAY\_STATUS(ws, i, excludedSheets)

Case "3"

flag = SHMEM\_UPF(ws, i, includedSheets, makeChanges)

Case "4"

flag = SHMEM\_PG0\_DELAY(ws, i, includedSheets)

End Select

If flag = 0 Then

ws.Cells(i, notFoundColumn).Value = "Test ID not found"

tpTestsNotInRegression = tpTestsNotInRegression + 1

End If

nextRow:

Next i

' Process CSV data

Set csvWorkbook = Workbooks.Open(CSV\_FILE\_PATH)

With csvWorkbook.Worksheets(1)

Dim lastRowCSV As Long

lastRowCSV = .Cells(.Rows.Count, 7).End(xlUp).row

ws.Range(ws.Cells(statsStartRow, 2), ws.Cells(statsStartRow, startColumn + 1)).ClearContents

.Range("G1").Copy

ws.Cells(statsStartRow, 2).PasteSpecial Paste:=xlPasteValues

If lastRowCSV > 1 Then

.Range("G2:G" & lastRowCSV).Copy

ws.Cells(statsStartRow + 1, 2).PasteSpecial Paste:=xlPasteValues

End If

End With

Set wbRegression = Workbooks.Open(CSV\_FILE\_PATH)

Set testIDDict = CreateObject("Scripting.Dictionary")

With wbRegression.Sheets(1)

lastRowCSV = .Cells(.Rows.Count, "E").End(xlUp).row

regressionTotal = lastRowCSV - 1

' Count occurrences of each test ID

For i = 2 To lastRowCSV

testID = Trim(.Cells(i, "E").Value)

If testID <> "" Then

If testIDDict.exists(testID) Then

testIDDict(testID) = testIDDict(testID) + 1

Else

testIDDict.Add testID, 1

End If

End If

Next i

' Output only the duplicate test IDs

OutputRow = statsStartRow + 1

With ws

For Each key In testIDDict.keys

If testIDDict(key) > 1 Then

.Cells(OutputRow, duplicateTestinRegression).Value = key

OutputRow = OutputRow + 1

End If

Next key

End With

With ws

.Cells(1, 1).Value = Choose(Val(userChoice), \_

"CHIP\_TOP Coverage", "CHIP\_TOP PG0 Coverage", "SH\_MEM Coverage", "SH\_MEM PG0 Coverage") & \_

" (" & Format(Date, "DD-MMM-YYYY") & ")"

End With

' Process regression data

Dim LastRowTestPlan As Long, LastRowRegression As Long

Dim TestPlanIDs As Range, RegressionIDs As Range

Dim RegressionID As Range

LastRowTestPlan = ws.Cells(ws.Rows.Count, Choose(Val(userChoice), "H", "G", "G", "F")).End(xlUp).row

LastRowRegression = csvWorkbook.Sheets(1).Cells(csvWorkbook.Sheets(1).Rows.Count, "E").End(xlUp).row

Set TestPlanIDs = ws.Range(ws.Cells(statsStartRow, Choose(Val(userChoice), 8, 7, 7, 6)), ws.Cells(LastRowTestPlan, Choose(Val(userChoice), 8, 7, 7, 6)))

Set RegressionIDs = csvWorkbook.Sheets(1).Range("E2:E" & LastRowRegression)

OutputRow = statsStartRow + 1

For Each RegressionID In RegressionIDs

If IsError(Application.Match(RegressionID.Value, TestPlanIDs, 0)) Then

ws.Cells(OutputRow, regressionMissingTests).Value = RegressionID.Value

OutputRow = OutputRow + 1

End If

Next RegressionID

' Initialize Worksheet

Set colMDict = CreateObject("Scripting.Dictionary")

With ws

lastRowM = .Cells(.Rows.Count, notFoundColumn - 1).End(xlUp).row

For i = statsStartRow + 1 To lastRowM

testIDM = Trim(.Cells(i, notFoundColumn - 1).Value)

If testIDM <> "" Then

If colMDict.exists(testIDM) Then

colMDict(testIDM) = colMDict(testIDM) + 1

Else

colMDict.Add testIDM, 1

End If

End If

Next i

OutputRowY = statsStartRow + 1

For Each key In colMDict.keys

If colMDict(key) > 1 Then

.Cells(OutputRowY, duplicateTestinTp).Value = key

OutputRowY = OutputRowY + 1

End If

Next key

End With

End With

lastRow = ws.Cells(ws.Rows.Count, notFoundColumn - 1).End(xlUp).row

Dim dict As Object

Set dict = CreateObject("Scripting.Dictionary")

For i = statsStartRow + 1 To lastRow

Dim keyValue As String

keyValue = Trim(ws.Cells(i, notFoundColumn - 1).Value)

If keyValue <> "" Then

If Not dict.exists(keyValue) Then

dict.Add keyValue, i

End If

End If

Next i

' Find last row with data in column X

Dim lastRowX As Long

lastRowX = ws.Cells(ws.Rows.Count, duplicateTestinRegression).End(xlUp).row

' Compare and copy data

For i = 2 To lastRowX ' Assuming row 1 is header

Dim compareValue As String

compareValue = Trim(ws.Cells(i, duplicateTestinRegression).Value)

' If value exists in dictionary (was found in Column L)

If compareValue <> "" And dict.exists(compareValue) Then

' Get the row number from dictionary

Dim matchRow As Long

matchRow = dict(compareValue)

' Copy corresponding data from Column N to Column Y

ws.Cells(i, duplicateTestinRegression + 1).Value = ws.Cells(matchRow, notFoundColumn + 1).Value

End If

Next i

With ws

.Columns("A").ColumnWidth = 17

.Columns("B").ColumnWidth = 22

.Columns("C").ColumnWidth = 12

.Columns(SVtestNameCol).ColumnWidth = 23

.Columns(notFoundColumn - 1).ColumnWidth = 28

.Columns(notFoundColumn).ColumnWidth = 18

.Columns(notFoundColumn + 1).ColumnWidth = 10

.Columns(notFoundColumn + 2).ColumnWidth = 33

.Columns(notFoundColumn + 3).ColumnWidth = 11

.Columns(notFoundColumn + 4).ColumnWidth = 5

.Columns(notFoundColumn + 5).ColumnWidth = 6

.Columns(regressionMissingTests).ColumnWidth = 28

.Columns(regressionMissingTests + 1).ColumnWidth = 11

.Columns(regressionMissingTests + 2).ColumnWidth = 33

.Columns(regressionMissingTests + 3).ColumnWidth = 11

.Columns(regressionMissingTests + 4).ColumnWidth = 5

.Columns(duplicateTestinRegression).ColumnWidth = 28

.Columns(duplicateTestinRegression + 1).ColumnWidth = 11

.Columns(duplicateTestinRegression + 2).ColumnWidth = 33

.Columns(duplicateTestinRegression + 3).ColumnWidth = 11

.Columns(duplicateTestinTp).ColumnWidth = 33

.Columns(duplicateTestinTp + 2).ColumnWidth = 33

.Cells(statsStartRow, 1).Value = Choose(Val(userChoice), \_

"CHIP\_TOP Coverage", "CHIP\_TOP PG0 Coverage", "SH\_MEM Coverage", "SH\_MEM PG0 Coverage") & \_

" (" & Format(Date, "DD-MMM-YYYY") & ")"

.Cells(statsStartRow, 2).Value = "Regression Sheets"

.Cells(statsStartRow, notFoundColumn).Value = "TP tests missing from regression"

.Cells(statsStartRow, notFoundColumn + 2).Value = "Comment"

.Cells(statsStartRow, notFoundColumn + 3).Value = "Status"

.Cells(statsStartRow, regressionMissingTests - 1).Value = "Sr.No"

.Cells(statsStartRow, regressionMissingTests).Value = "Regression tests missing in TP"

.Cells(statsStartRow, regressionMissingTests + 1).Value = "Owner"

.Cells(statsStartRow, regressionMissingTests + 2).Value = "Comment"

.Cells(statsStartRow, regressionMissingTests + 3).Value = "Status"

.Cells(statsStartRow, regressionMissingTests + 5).Value = "Duplicate Test ID's in regression"

.Cells(statsStartRow, regressionMissingTests + 6).Value = "Owner"

.Cells(statsStartRow, regressionMissingTests + 7).Value = "Comment"

.Cells(statsStartRow, regressionMissingTests + 8).Value = "Status"

.Cells(statsStartRow, regressionMissingTests + 9).Value = "Duplicate Test ID's in Testplan"

.Cells(statsStartRow, regressionMissingTests + 10).Value = "Owner"

.Cells(statsStartRow, regressionMissingTests + 11).Value = "Comment"

.Cells(statsStartRow, regressionMissingTests + 12).Value = "Status"

With ws

missingInTP = .Cells(.Rows.Count, regressionMissingTests).End(xlUp).row - statsStartRow

' Update missingInTP count in statistics

If missingInTP > 0 Then

.Cells(statsStartRow + 12, 1).Value = missingInTP

Else

.Cells(statsStartRow + 12, 1).Value = 0

End If

' Get last row with data in regression missing tests column

lastRowN = .Cells(.Rows.Count, regressionMissingTests).End(xlUp).row

' Add sequential numbers for visible rows with missing tests

counter = 1

For i = statsStartRow + 1 To lastRowN

If Not .Rows(i).Hidden Then

If Not IsEmpty(.Cells(i, regressionMissingTests).Value) Then

.Cells(i, regressionMissingTests - 1).Value = counter

counter = counter + 1

End If

End If

Next i

End With

With ws

.Cells(statsStartRow + 1, 1).Value = "Testplan Total Tests"

.Cells(statsStartRow + 2, 1).Value = testplanTotal

.Cells(statsStartRow + 3, 1).Value = "Regression Total Tests"

.Cells(statsStartRow + 4, 1).Value = regressionTotal

.Cells(statsStartRow + 5, 1).Value = "Testplan vs. Regression coverage"

If testplanTotal > 0 Then

coverage = ((testplanTotal - tpTestsNotInRegression) / testplanTotal) \* 100

Else

coverage = 0

End If

.Cells(statsStartRow + 6, 1).Value = Format(coverage, "0.00") & "%"

.Cells(statsStartRow + 7, 1).Value = "Regression vs. Testplan coverage"

If regressionTotal > 0 Then

coverage = ((regressionTotal - missingInTP) / regressionTotal) \* 100

Else

coverage = 0

End If

.Cells(statsStartRow + 8, 1).Value = Format(coverage, "0.00") & "%"

.Cells(statsStartRow + 9, 1).Value = "TP tests not in regression:"

.Cells(statsStartRow + 10, 1).Value = tpTestsNotInRegression

.Cells(statsStartRow + 11, 1).Value = "Regression tests not in TP"

.Cells(statsStartRow + 13, 1).Value = "Duplicate tests in regression"

duplicateTests = 0

Dim lastDuplicateRow As Long

lastDuplicateRow = .Cells(.Rows.Count, duplicateTestinRegression).End(xlUp).row

' Count visible rows only

For i = statsStartRow + 1 To lastDuplicateRow

If Not .Rows(i).Hidden Then

If Not IsEmpty(.Cells(i, duplicateTestinRegression).Value) Then

duplicateTests = duplicateTests + 1

End If

End If

Next i

' Update duplicate tests count

.Cells(statsStartRow + 14, 1).Value = duplicateTests

End With

' Header Formatting

With .Range(.Cells(statsStartRow, 1), .Cells(statsStartRow, lastHeaderColumn))

.HorizontalAlignment = xlCenter

.VerticalAlignment = xlCenter

.WrapText = True

.Font.Bold = True

.Interior.Color = RGB(200, 200, 200) ' Darker grey background

End With

' Clear all existing borders

.Cells.Borders.LineStyle = xlNone

' First border: Column A to notFoundColumn+3

lastRowA = .Cells(.Rows.Count, "C").End(xlUp).row

If lastRowA > statsStartRow Then

.Range(.Cells(statsStartRow, 1), .Cells(lastRowA, notFoundColumn + 3)).Borders.LineStyle = xlContinuous

End If

' Second border: regressionMissingTests-1 to regressionMissingTests+3

lastRowM = .Cells(.Rows.Count, regressionMissingTests).End(xlUp).row

If lastRowM > statsStartRow Then

.Range(.Cells(statsStartRow, regressionMissingTests - 1), .Cells(lastRowM, regressionMissingTests + 3)).Borders.LineStyle = xlContinuous

End If

' Third border: duplicateTestinRegression to duplicateTestinRegression+7

lastRowS = .Cells(.Rows.Count, duplicateTestinRegression).End(xlUp).row

If lastRowS > statsStartRow Then

.Range(.Cells(statsStartRow, duplicateTestinRegression), .Cells(lastRowS, duplicateTestinRegression + 7)).Borders.LineStyle = xlContinuous

End If

lastRow = .Cells(.Rows.Count, "C").End(xlUp).row

lastCol = .Cells(1, .Columns.Count).End(xlToLeft).Column

' wrap all cells from row 2 to last row

Set rng = .Range(.Cells(statsStartRow + 1, 1), .Cells(lastRow, lastCol))

rng.WrapText = True

.Columns("A").HorizontalAlignment = xlCenter

.Columns("A").VerticalAlignment = xlCenter

.Columns(regressionMissingTests - 1).HorizontalAlignment = xlCenter

.Columns(regressionMissingTests - 1).VerticalAlignment = xlCenter

.Columns("E").WrapText = False

.Columns("G").WrapText = False

.Columns("F").WrapText = False

End With

csvWorkbook.Close SaveChanges:=False

' Show completion message

MsgBox "Process completed successfully for " & sheetName, vbInformation

End Sub

Private Function UPF\_STATUS(ws As Worksheet, currentRow As Long, excludedSheets As Variant, makeChanges As Boolean) As Integer

Dim targetSheet As Worksheet

Dim lastRowWs As Long

Dim j As Long

For Each targetSheet In Worksheets

If Not IsInExcludedSheet(targetSheet.Name, excludedSheets) Then

lastRowWs = targetSheet.Cells(targetSheet.Rows.Count, 4).End(xlUp).row

For j = 6 To lastRowWs

If targetSheet.Cells(j, 4).Value = ws.Cells(currentRow, 8).Value Then

If makeChanges Then

targetSheet.Cells(j, 6).Value = targetSheet.Cells(j, 5).Value

End If

targetSheet.Cells(j, 5).Value = ws.Cells(currentRow, 3).Value

targetSheet.Cells(j, 8).Value = ws.Cells(currentRow, 5).Value

UPF\_STATUS = 1

Exit Function

End If

Next j

End If

Next targetSheet

UPF\_STATUS = 0

End Function

Private Function PG0\_DELAY\_STATUS(ws As Worksheet, currentRow As Long, excludedSheets As Variant) As Integer

Dim targetSheet As Worksheet

Dim lastRowWs As Long

Dim j As Long

For Each targetSheet In Worksheets

If Not IsInExcludedSheet(targetSheet.Name, excludedSheets) Then

lastRowWs = targetSheet.Cells(targetSheet.Rows.Count, 4).End(xlUp).row

For j = 6 To lastRowWs

If targetSheet.Cells(j, 4).Value = ws.Cells(currentRow, 7).Value Then

targetSheet.Cells(j, 7).Value = ws.Cells(currentRow, 3).Value

PG0\_DELAY\_STATUS = 1

Exit Function

End If

Next j

End If

Next targetSheet

PG0\_DELAY\_STATUS = 0

End Function

Private Function SHMEM\_UPF(ws As Worksheet, currentRow As Long, includedSheets As Variant, makeChanges As Boolean) As Integer

Dim targetSheet As Worksheet

Dim lastRowWs As Long

Dim j As Long

For Each targetSheet In Worksheets

' Use IsSheetIncluded function to check sheet

If IsSheetIncluded(targetSheet.Name, includedSheets) Then

' Process the included sheet

lastRowWs = targetSheet.Cells(targetSheet.Rows.Count, 4).End(xlUp).row

For j = 6 To lastRowWs

If targetSheet.Cells(j, 3).Value = ws.Cells(currentRow, 7).Value Then

' Always update, even if current cell is empty

If makeChanges Then

targetSheet.Cells(j, 5).Value = targetSheet.Cells(j, 4).Value

End If

targetSheet.Cells(j, 4).Value = ws.Cells(currentRow, 3).Value

targetSheet.Cells(j, 7).Value = ws.Cells(currentRow, 5).Value

SHMEM\_UPF = 1

Exit Function

End If

Next j

End If

Next targetSheet

SHMEM\_UPF = 0

End Function

Private Function SHMEM\_PG0\_DELAY(ws As Worksheet, currentRow As Long, includedSheets As Variant) As Integer

Dim targetSheet As Worksheet

Dim lastRowWs As Long

Dim j As Long

For Each targetSheet In Worksheets

' Use IsSheetIncluded function to check sheet

If IsSheetIncluded(targetSheet.Name, includedSheets) Then

' Process the included sheet

lastRowWs = targetSheet.Cells(targetSheet.Rows.Count, 4).End(xlUp).row

For j = 6 To lastRowWs

' Compare Test ID

If targetSheet.Cells(j, 3).Value = ws.Cells(currentRow, 6).Value Then

targetSheet.Cells(j, 6).Value = ws.Cells(currentRow, 3).Value

SHMEM\_PG0\_DELAY = 1

Exit Function

End If

Next j

End If

Next targetSheet

SHMEM\_PG0\_DELAY = 0

End Function

Private Function IsInExcludedSheet(sheetName As String, excludedSheets As Variant) As Boolean

Dim i As Integer

For i = LBound(excludedSheets) To UBound(excludedSheets)

If sheetName = excludedSheets(i) Then

IsInExcludedSheet = True

Exit Function

End If

Next i

IsInExcludedSheet = False

End Function

Private Sub SDF\_STATUS(sheetType As String)

' Variable declarations

Dim makeChangestyp\_max As Boolean

Dim makeChangestyp\_min As Boolean

Dim makeChangesmaxmax As Boolean

Dim makeChangesminmin As Boolean

Dim excludedSheets As Variant

Dim includedSheets As Variant

Dim wsMain As Worksheet

Dim lastRow As Long

Dim lastCol As Long

Dim rng As Range

Dim testplanTotal As Long

Dim regressionTotal As Long

Dim CSV\_FILE\_PATHS As Variant

Dim filePath As Variant

Dim fileIndex As Long

Dim wbRegression As Workbook

Dim CSV\_FILE\_PATH As String

Dim mainLastRow As Long

Dim regressionLastRow As Long

Dim mainTestIDs As Range

Dim testIDDict As Object

Dim lastRowCSV As Long

Dim i As Long

Dim testID As String

Dim OutputRow As Long

Dim LastRowTestPlan As Long

Dim LastRowRegression As Long

Dim TestPlanIDs As Range

Dim RegressionIDs As Range

Dim RegressionID As Range

Dim tpTestsNotInRegression As Long

Dim lastRowT As Long

Dim missingInTP As Long

Dim coverage As Double

Dim ws As Worksheet

Dim sheetExcluded As Boolean

Dim sheetIncluded As Boolean

Dim compareLastRow As Long

Dim compareTestIDCol As Range

Dim cell As Range

Dim found As Range

Dim lastRowA As Long

Dim key As Variant

Dim colMDict As Object

Dim lastRowM As Long

Dim testIDM As String

Dim OutputRowY As Long

Dim dict As Object

Dim keyValue As String

Dim lastRowX As Long

Dim compareValue As String

Dim matchRow As Long

Dim duplicateTests As Long

Dim userInput As String

Dim statsStartRow As Long

Dim visibleRowCount As Long

On Error GoTo ErrorHandler

' Initialize variables

makeChangestyp\_max = True

makeChangestyp\_min = True

makeChangesmaxmax = True

makeChangesminmin = True

tpTestsNotInRegression = 0

visibleRowCount = 0

Application.ScreenUpdating = False

Application.Calculation = xlCalculationManual

' Define excluded/included sheets based on type

If sheetType = "1" Then

excludedSheets = Array("TP\_COVERAGE\_UPF", "TP\_COVERAGE\_SHMEM\_SDF", \_

"TP\_COVERAGE\_CT\_SDF", "TP\_COVERAGE\_CT\_TYP\_MAX", \_

"TP\_COVERAGE\_CT\_TYP\_MIN", "TP\_COVERAGE\_SHMEM\_TYP\_MAX", \_

"TP\_COVERAGE\_SHMEM\_TYP\_MIN", "TP\_COVERAGE\_CT\_TYP\_MAX\_MAX", \_

"TP\_COVERAGE\_CT\_TYP\_MIN\_MIN", "TP\_COVERAGE\_SHMEM\_TYP\_MAX\_MAX", \_

"TP\_COVERAGE\_SHMEM\_TYP\_MIN\_MIN", "TP\_COVERAGE\_PG0\_DELAY", \_

"TP\_COVERAGE\_SH\_MEM\_UPF", "TP\_COVERAGE\_SH\_MEM\_PG0\_DELAY", \_

"TP\_COVERAGE\_SHMEM\_SDF\_TYP\_MIN", "TP\_COVERAGE\_SHMEM\_SDF\_TYP\_MAX", \_

"TP\_COVERAGE\_CT\_SDF\_TYP\_MIN", "TP\_COVERAGE\_CT\_SDF\_TYP\_MAX", \_

"TP\_COVERAGE\_GLS\_SDF", "GLS\_SDF\_SANITY\_HP", "CHIP\_TOP", "MBIST", \_

"PINMUX", "apb\_i2c", "OSPI\_SS", "JPEG\_SS", "MRAM\_SS", "ZAPHOD\_SS", \_

"ISP\_SS", "SHARED\_MEM\_SS", "MRAM\_RC\_SS", "CVM\_OCVM\_SS", "CPI\_ISP\_SS", \_

"MRAM\_SS\_OLD", "OSPI\_SS\_plan", "JPEG\_SS\_OLD")

Set wsMain = ThisWorkbook.Sheets("TP\_COVERAGE\_CT\_SDF")

Else

includedSheets = GetSheetsEndingWith("\_SS")

Set wsMain = ThisWorkbook.Sheets("TP\_COVERAGE\_SHMEM\_SDF")

End If

' Find first visible row

statsStartRow = wsMain.UsedRange.row

While wsMain.Rows(statsStartRow).Hidden

statsStartRow = statsStartRow + 1

Wend

' Format main worksheet - only visible rows

With wsMain

lastRow = .Cells(.Rows.Count, "C").End(xlUp).row

While lastRow > statsStartRow And .Rows(lastRow).Hidden

lastRow = lastRow - 1

Wend

lastCol = .Cells(statsStartRow, .Columns.Count).End(xlToLeft).Column

For i = statsStartRow + 1 To lastRow

If Not .Rows(i).Hidden Then

.Range(.Cells(i, 1), .Cells(i, lastCol)).WrapText = False

visibleRowCount = visibleRowCount + 1

End If

Next i

End With

' Process duplicate Test IDs - only visible rows

Set colMDict = CreateObject("Scripting.Dictionary")

With wsMain

lastRowM = .Cells(.Rows.Count, IIf(sheetType = "1", "M", "L")).End(xlUp).row

For i = statsStartRow + 1 To lastRowM

If Not .Rows(i).Hidden Then

testIDM = Trim(.Cells(i, IIf(sheetType = "1", "M", "L")).Value)

If testIDM <> "" Then

If colMDict.exists(testIDM) Then

colMDict(testIDM) = colMDict(testIDM) + 1

Else

colMDict.Add testIDM, 1

End If

End If

End If

Next i

OutputRowY = statsStartRow + 1

For Each key In colMDict.keys

If colMDict(key) > 1 Then

.Cells(OutputRowY, IIf(sheetType = "1", 29, 28)).Value = key

OutputRowY = OutputRowY + 1

End If

Next key

End With

' Define CSV file paths

If sheetType = "1" Then

CSV\_FILE\_PATHS = Array( \_

"<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_CT_TYP_MAX.csv>", \_

"<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_CT_TYP_MIN.csv>", \_

"<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_CT_TYP_MAX.csv>", \_

"<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_CT_TYP_MIN.csv>")

Else

CSV\_FILE\_PATHS = Array( \_

"<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_SHMEM_TYP_MAX.csv>", \_

"<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_SHMEM_TYP_MIN.csv>", \_

"<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_SHMEM_TYP_MAX_MAX.csv>", \_

"<https://alifsemi.sharepoint.com\sites\dv_team\Shared%20Documents\Docs\Eagle_A0\TestPlan\REGRESSION_SHEETS\Regression_SHMEM_TYP_MIN_MIN.csv>")

End If

' Process regression files

For fileIndex = LBound(CSV\_FILE\_PATHS) To UBound(CSV\_FILE\_PATHS)

On Error Resume Next

Set wbRegression = Workbooks.Open(CSV\_FILE\_PATHS(fileIndex))

On Error GoTo 0

If wbRegression Is Nothing Then

MsgBox "Regression CSV file not found at: " & CSV\_FILE\_PATHS(fileIndex), vbCritical

GoTo NextFile

End If

With wbRegression.Sheets(1)

If Not IsEmpty(.Cells(2, 7)) Then

.Cells(2, 7).Copy

wsMain.Cells(fileIndex + statsStartRow + 1, 2).PasteSpecial xlPasteValues

End If

End With

wbRegression.Close False

NextFile:

Next fileIndex

' Process main regression file

CSV\_FILE\_PATH = CSV\_FILE\_PATHS(0)

On Error Resume Next

Set wbRegression = Workbooks.Open(CSV\_FILE\_PATH)

On Error GoTo 0

If wbRegression Is Nothing Then

MsgBox "Main regression CSV file not found at: " & CSV\_FILE\_PATH, vbCritical

GoTo CleanupAndExit

End If

' Process Test ID not found checks - only for visible rows

With wsMain

tpTestsNotInRegression = 0

Dim checkColumn As Long

Dim notFoundColumn As Long

' Set columns based on sheet type

checkColumn = IIf(sheetType = "1", 3, 3) ' Column C

notFoundColumn = IIf(sheetType = "1", 14, 13) ' Column N or M

' Process each visible row

For i = statsStartRow + 1 To lastRow

If Not .Rows(i).Hidden Then

' Check if test ID is empty or missing

If Trim(.Cells(i, checkColumn).Value) = "" Then

.Cells(i, notFoundColumn).Value = "Test ID not found"

tpTestsNotInRegression = tpTestsNotInRegression + 1

Else

' Check if test ID exists in regression

Dim foundInRegression As Boolean

foundInRegression = False

' Get the test ID from the appropriate column

testID = Trim(.Cells(i, IIf(sheetType = "1", "M", "L")).Value)

If testID <> "" Then

With wbRegression.Sheets(1)

Dim regLastRow As Long

regLastRow = .Cells(.Rows.Count, "E").End(xlUp).row

' Search in regression sheet

For j = 2 To regLastRow

If Trim(.Cells(j, "E").Value) = testID Then

foundInRegression = True

Exit For

End If

Next j

End With

If Not foundInRegression Then

wsMain.Cells(i, notFoundColumn).Value = "Test ID not found"

tpTestsNotInRegression = tpTestsNotInRegression + 1

End If

Else

' If test ID column is empty

.Cells(i, notFoundColumn).Value = "Test ID not found"

tpTestsNotInRegression = tpTestsNotInRegression + 1

End If

End If

End If

Next i

' Update the Test ID not found count in statistics

.Cells(statsStartRow + 10, 1).Value = tpTestsNotInRegression

End With

' Process regression data and duplicates

Set testIDDict = CreateObject("Scripting.Dictionary")

With wbRegression.Sheets(1)

lastRowCSV = .Cells(.Rows.Count, "E").End(xlUp).row

For i = 2 To lastRowCSV

testID = Trim(.Cells(i, "E").Value)

If testID <> "" Then

If testIDDict.exists(testID) Then

testIDDict(testID) = testIDDict(testID) + 1

Else

testIDDict.Add testID, 1

End If

End If

Next i

' Output duplicate regression tests

OutputRow = statsStartRow + 1

For Each key In testIDDict.keys

If testIDDict(key) > 1 Then

wsMain.Cells(OutputRow, IIf(sheetType = "1", 25, 24)).Value = key

OutputRow = OutputRow + 1

End If

Next key

' Process missing tests between TestPlan and Regression

LastRowTestPlan = wsMain.Cells(wsMain.Rows.Count, IIf(sheetType = "1", "D", "C")).End(xlUp).row

LastRowRegression = .Cells(.Rows.Count, "E").End(xlUp).row

' Get visible TestPlan IDs

Dim visibleTestPlanIDs As Collection

Set visibleTestPlanIDs = New Collection

For i = statsStartRow + 1 To LastRowTestPlan

If Not wsMain.Rows(i).Hidden Then

testID = Trim(wsMain.Cells(i, IIf(sheetType = "1", "M", "L")).Value)

If testID <> "" Then

On Error Resume Next

visibleTestPlanIDs.Add testID, testID

On Error GoTo 0

End If

End If

Next i

' Check regression tests against visible TestPlan IDs

OutputRow = statsStartRow + 1

For i = 2 To LastRowRegression

testID = Trim(.Cells(i, "E").Value)

If testID <> "" Then

On Error Resume Next

visibleTestPlanIDs.Item testID

If Err.Number <> 0 Then

' Test not found in TestPlan

wsMain.Cells(OutputRow, IIf(sheetType = "1", 20, 19)).Value = testID

OutputRow = OutputRow + 1

End If

On Error GoTo 0

End If

Next i

End With

' Update serial numbers for visible rows

With wsMain

lastRowT = .Cells(.Rows.Count, IIf(sheetType = "1", "T", "S")).End(xlUp).row

Dim serialNumber As Long

serialNumber = 1

For i = statsStartRow + 1 To lastRowT

If Not .Rows(i).Hidden Then

If Not IsEmpty(.Cells(i, IIf(sheetType = "1", "T", "S")).Value) Then

.Cells(i, IIf(sheetType = "1", "S", "R")).Value = serialNumber

serialNumber = serialNumber + 1

End If

End If

Next i

End With

' Set up dictionary for matching between sheets

Set dict = CreateObject("Scripting.Dictionary")

With wsMain

lastRow = .Cells(.Rows.Count, IIf(sheetType = "1", "M", "L")).End(xlUp).row

For i = statsStartRow + 1 To lastRow

If Not .Rows(i).Hidden Then

keyValue = Trim(.Cells(i, IIf(sheetType = "1", "M", "L")).Value)

If keyValue <> "" Then

If Not dict.exists(keyValue) Then

dict.Add keyValue, i

End If

End If

End If

Next i

' Match and update values

lastRowX = .Cells(.Rows.Count, IIf(sheetType = "1", "Y", "X")).End(xlUp).row

For i = statsStartRow + 1 To lastRowX

If Not .Rows(i).Hidden Then

compareValue = Trim(.Cells(i, IIf(sheetType = "1", "Y", "X")).Value)

If compareValue <> "" And dict.exists(compareValue) Then

matchRow = dict(compareValue)

.Cells(i, IIf(sheetType = "1", "Z", "Y")).Value = \_

.Cells(matchRow, IIf(sheetType = "1", "O", "N")).Value

End If

End If

Next i

End With

With wsMain

Dim lastDuplicateRow As Long

Dim visibleDuplicateCount As Long

' Get last row in duplicate column

lastDuplicateRow = .Cells(.Rows.Count, IIf(sheetType = "1", 25, 24)).End(xlUp).row

' Count only visible rows

visibleDuplicateCount = 0

For i = statsStartRow + 1 To lastDuplicateRow

If Not .Rows(i).Hidden Then

If Not IsEmpty(.Cells(i, IIf(sheetType = "1", 25, 24)).Value) Then

visibleDuplicateCount = visibleDuplicateCount + 1

End If

End If

Next i

' Update duplicate count

duplicateTests = visibleDuplicateCount

' Update statistics

.Cells(statsStartRow + 14, 1).Value = duplicateTests

End With

' Update related sheets

For Each ws In ThisWorkbook.Worksheets

If sheetType = "1" Then

' Process CT\_SDF sheets

sheetExcluded = False

For i = LBound(excludedSheets) To UBound(excludedSheets)

If ws.Name = excludedSheets(i) Then

sheetExcluded = True

Exit For

End If

Next i

If Not sheetExcluded And ws.Name <> wsMain.Name Then

compareLastRow = ws.Cells(ws.Rows.Count, "D").End(xlUp).row

For i = statsStartRow + 1 To compareLastRow

If Not ws.Rows(i).Hidden Then

testID = Trim(ws.Cells(i, "D").Value)

If testID <> "" And dict.exists(testID) Then

matchRow = dict(testID)

If makeChangestyp\_max Then ws.Cells(i, "K").Value = ws.Cells(i, "J").Value

ws.Cells(i, "J").Value = wsMain.Cells(matchRow, "C").Value

If makeChangestyp\_min Then ws.Cells(i, "M").Value = ws.Cells(i, "L").Value

ws.Cells(i, "L").Value = wsMain.Cells(matchRow, "E").Value

If makeChangesmaxmax Then ws.Cells(i, "O").Value = ws.Cells(i, "N").Value

ws.Cells(i, "N").Value = wsMain.Cells(matchRow, "G").Value

If makeChangesminmin Then ws.Cells(i, "Q").Value = ws.Cells(i, "P").Value

ws.Cells(i, "P").Value = wsMain.Cells(matchRow, "I").Value

End If

End If

Next i

End If

Else

' Process SHMEM\_SDF sheets

sheetIncluded = False

For i = LBound(includedSheets) To UBound(includedSheets)

If ws.Name = includedSheets(i) Then

sheetIncluded = True

Exit For

End If

Next i

If sheetIncluded Then

compareLastRow = ws.Cells(ws.Rows.Count, "C").End(xlUp).row

For i = statsStartRow + 1 To compareLastRow

If Not ws.Rows(i).Hidden Then

testID = Trim(ws.Cells(i, "C").Value)

If testID <> "" And dict.exists(testID) Then

matchRow = dict(testID)

If makeChangestyp\_max Then ws.Cells(i, "J").Value = ws.Cells(i, "I").Value

ws.Cells(i, "I").Value = wsMain.Cells(matchRow, "C").Value

If makeChangestyp\_min Then ws.Cells(i, "L").Value = ws.Cells(i, "K").Value

ws.Cells(i, "K").Value = wsMain.Cells(matchRow, "E").Value

If makeChangesmaxmax Then ws.Cells(i, "N").Value = ws.Cells(i, "M").Value

ws.Cells(i, "M").Value = wsMain.Cells(matchRow, "G").Value

If makeChangesminmin Then ws.Cells(i, "P").Value = ws.Cells(i, "O").Value

ws.Cells(i, "O").Value = wsMain.Cells(matchRow, "I").Value

End If

End If

Next i

End If

End If

Next ws

' Calculate metrics and update worksheet

With wsMain

' Add headers at first visible row

.Cells(statsStartRow, 1).Value = "SDF " & IIf(sheetType = "1", "CT", "SHMEM") & \_

" Coverage)"

.Cells(statsStartRow, 2).Value = "Regression Sheets"

.Cells(statsStartRow, IIf(sheetType = "1", 14, 13)).Value = "TP tests missing from regression"

.Cells(statsStartRow, IIf(sheetType = "1", 16, 15)).Value = "Comment"

.Cells(statsStartRow, IIf(sheetType = "1", 17, 16)).Value = "Status"

.Cells(statsStartRow, IIf(sheetType = "1", 19, 18)).Value = "Sr.No"

.Cells(statsStartRow, IIf(sheetType = "1", 20, 19)).Value = "Regression tests missing in TP"

.Cells(statsStartRow, IIf(sheetType = "1", 21, 20)).Value = "Owner"

.Cells(statsStartRow, IIf(sheetType = "1", 22, 21)).Value = "Comment"

.Cells(statsStartRow, IIf(sheetType = "1", 23, 22)).Value = "Status"

.Cells(statsStartRow, IIf(sheetType = "1", 25, 24)).Value = "Duplicate Test ID's in regression"

.Cells(statsStartRow, IIf(sheetType = "1", 26, 25)).Value = "Owner"

.Cells(statsStartRow, IIf(sheetType = "1", 27, 26)).Value = "Comment"

.Cells(statsStartRow, IIf(sheetType = "1", 28, 27)).Value = "Status"

.Cells(statsStartRow, IIf(sheetType = "1", 29, 28)).Value = "Duplicate Test ID's in TestPlan"

' Update statistics

.Cells(statsStartRow + 1, 1).Value = "Testplan Total Tests"

.Cells(statsStartRow + 2, 1).Value = visibleRowCount

.Cells(statsStartRow + 3, 1).Value = "Regression Total Tests"

' Calculate regression total from visible rows

regressionTotal = 0

With wbRegression.Sheets(1)

lastRowCSV = .Cells(.Rows.Count, "E").End(xlUp).row

For i = 2 To lastRowCSV

If Not .Rows(i).Hidden Then

regressionTotal = regressionTotal + 1

End If

Next i

End With

.Cells(statsStartRow + 4, 1).Value = regressionTotal

' Calculate coverage percentages

missingInTP = .Cells(.Rows.Count, IIf(sheetType = "1", 20, 19)).End(xlUp).row - statsStartRow

If missingInTP > 0 Then

.Cells(statsStartRow + 12, 1).Value = missingInTP

Else

.Cells(statsStartRow + 12, 1).Value = 0

End If

' Update test coverage statistics

.Cells(statsStartRow + 5, 1).Value = "Testplan vs. Regression coverage"

If visibleRowCount > 0 Then

coverage = ((visibleRowCount - tpTestsNotInRegression) / visibleRowCount) \* 100

Else

coverage = 0

End If

.Cells(statsStartRow + 6, 1).Value = Format(coverage, "0.00") & "%"

.Cells(statsStartRow + 7, 1).Value = "Regression vs. Testplan coverage"

If regressionTotal > 0 Then

coverage = ((regressionTotal - missingInTP) / regressionTotal) \* 100

Else

coverage = 0

End If

.Cells(statsStartRow + 8, 1).Value = Format(coverage, "0.00") & "%"

' Update missing tests information

.Cells(statsStartRow + 9, 1).Value = "TP tests not in regression:"

.Cells(statsStartRow + 10, 1).Value = tpTestsNotInRegression

.Cells(statsStartRow + 11, 1).Value = "Regression tests not in TP"

.Cells(statsStartRow + 13, 1).Value = "Duplicate tests in regression"

' Format header row

With .Range(.Cells(statsStartRow, 1), .Cells(statsStartRow, IIf(sheetType = "1", 31, 30)))

.HorizontalAlignment = xlCenter

.VerticalAlignment = xlCenter

.WrapText = True

.Font.Bold = True

.Interior.Color = RGB(200, 200, 200)

End With

' Add borders to visible range

lastRowA = .Cells(.Rows.Count, "C").End(xlUp).row

.Range(.Cells(statsStartRow, 1), .Cells(lastRowA, IIf(sheetType = "1", "AE", "AD"))).Borders.LineStyle = xlContinuous

End With

' Format worksheet

If sheetType = "1" Then

With wsMain

' Format worksheet with +1 width

.Columns("A").ColumnWidth = 17

.Columns("B").ColumnWidth = 22

.Columns("C:J").ColumnWidth = 10

.Columns("K:L").ColumnWidth = 23

.Columns("M").ColumnWidth = 28

.Columns("N").ColumnWidth = 18

.Columns("O").ColumnWidth = 10

.Columns("P").ColumnWidth = 33

.Columns("Q").ColumnWidth = 11

.Columns("R").ColumnWidth = 5

.Columns("S").ColumnWidth = 5

.Columns("T").ColumnWidth = 28

.Columns("U").ColumnWidth = 11

.Columns("V").ColumnWidth = 33

.Columns("W").ColumnWidth = 11

.Columns("X").ColumnWidth = 5

.Columns("Y").ColumnWidth = 28

.Columns("Z").ColumnWidth = 11

.Columns("AA").ColumnWidth = 33

.Columns("AB").ColumnWidth = 11

.Columns("AC").ColumnWidth = 33

.Columns("AD").ColumnWidth = 11

.Columns("AE").ColumnWidth = 33

.Columns("AF").ColumnWidth = 11

.Columns("A").HorizontalAlignment = xlCenter

.Columns("A").VerticalAlignment = xlCenter

.Columns("K:L").WrapText = False

.Columns("M").WrapText = True

.Columns("A").WrapText = True

End With

ElseIf sheetType = "2" Then

With wsMain

' Original formatting

.Columns("A").ColumnWidth = 17

.Columns("B").ColumnWidth = 22

.Columns("C:J").ColumnWidth = 10

.Columns("K").ColumnWidth = 23

.Columns("L").ColumnWidth = 28

.Columns("M").ColumnWidth = 18

.Columns("N").ColumnWidth = 10

.Columns("O").ColumnWidth = 33

.Columns("P").ColumnWidth = 11

.Columns("Q").ColumnWidth = 5

.Columns("R").ColumnWidth = 5

.Columns("S").ColumnWidth = 28

.Columns("T").ColumnWidth = 11

.Columns("U").ColumnWidth = 33

.Columns("V").ColumnWidth = 11

.Columns("W").ColumnWidth = 5

.Columns("X").ColumnWidth = 28

.Columns("Y").ColumnWidth = 11

.Columns("Z").ColumnWidth = 33

.Columns("AA").ColumnWidth = 11

.Columns("AB").ColumnWidth = 33

.Columns("AC").ColumnWidth = 11

.Columns("AD").ColumnWidth = 33

.Columns("AE").ColumnWidth = 11

.Columns("A").HorizontalAlignment = xlCenter

.Columns("A").VerticalAlignment = xlCenter

.Columns("K").WrapText = False

.Columns("A").WrapText = True

.Columns("L").WrapText = True

End With

End If

CleanupAndExit:

Application.ScreenUpdating = True

Application.Calculation = xlCalculationAutomatic

MsgBox "Coverage analysis and update completed successfully!", vbInformation

Exit Sub

ErrorHandler:

Application.ScreenUpdating = True

Application.Calculation = xlCalculationAutomatic

If Not wbRegression Is Nothing Then wbRegression.Close False

MsgBox "An error occurred: " & Err.Description, vbCritical

End Sub

Public Function GetSheetsEndingWith(suffix As String) As Variant

Dim ws As Worksheet

Dim matchingSheets As Collection

Set matchingSheets = New Collection

For Each ws In ThisWorkbook.Worksheets

If Right(ws.Name, Len(suffix)) = suffix Then

matchingSheets.Add ws.Name

End If

Next ws

Dim result() As String

Dim i As Long

ReDim result(1 To matchingSheets.Count)

For i = 1 To matchingSheets.Count

result(i) = matchingSheets(i)

Next i

GetSheetsEndingWith = result

End Function

Private Function IsSheetIncluded(sheetName As String, includedSheets As Variant) As Boolean

Dim k As Long

For k = LBound(includedSheets) To UBound(includedSheets)

If sheetName = includedSheets(k) Then

IsSheetIncluded = True

Exit Function

End If

Next k

IsSheetIncluded = False

End Function