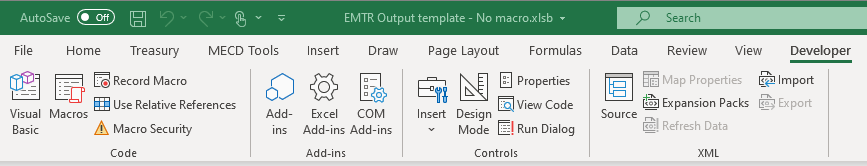
# Instructions for using macros

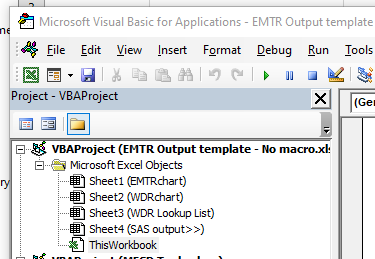
Macros will need to be added into two Excel workbooks. The *‘Public version CPS - Budget 2021-22.xlsx*’ workbook in the main CAPITA folder and the *‘EMTR Output template.xlsb*’ in the Cameo folder. The macros for each workbook can be found in the next section of this document.

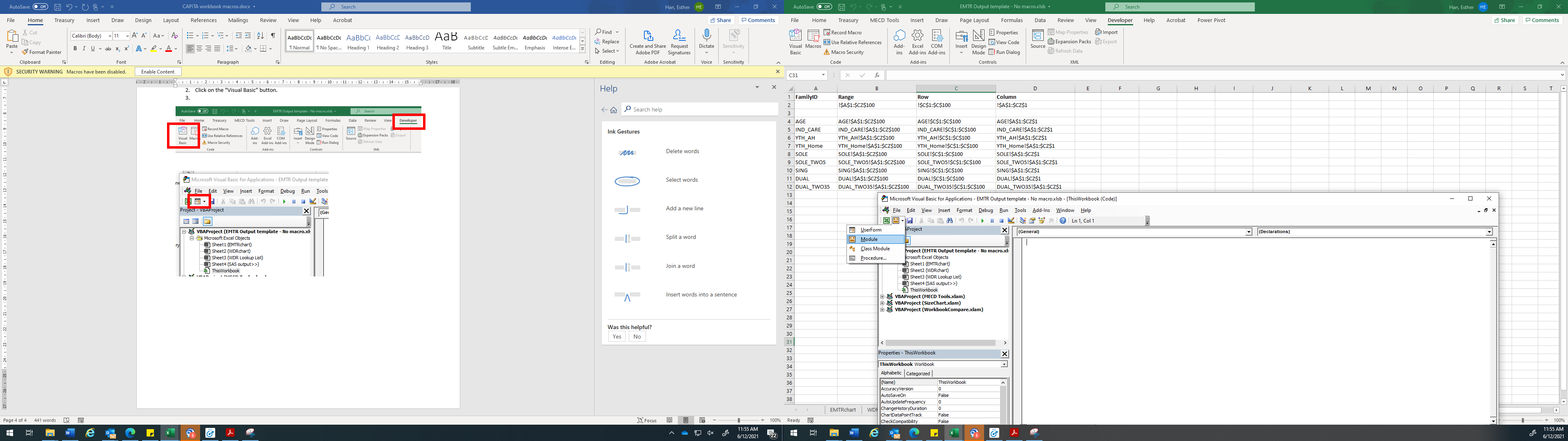
Use the following instructions to add the relevant macros into the individual Excel workbooks.

1. Go to “Developer” tab in the workbook.
2. Click on the “Visual Basic” button to open the Microsoft Visual Basic Editor.

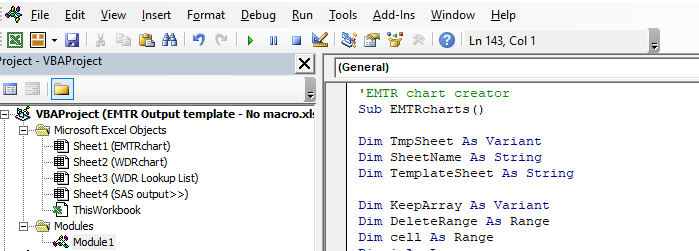


1. In the Microsoft Visual Basic Editor click on the modules drop down to add a module in the workbook.





1. Excel will automatically create a “Module 1”. You can either rename this or keep it. In Module 1 copy and paste the relevant macro code from the next section of this document (highlighted in yellow).



1. Save the changes and close the Microsoft Visual Basic Editor (for the Common Parameters Spreadsheet (CPS), make sure the spreadsheet is saved as a .xlsm or .xlsb to use Excel macros).

# CAPITA: Excel workbook macros

## Public version CPS - Budget 2021-22.xlsb

The following macros are for the Common Parameter spreadsheet (CPS).

'Replicate MROUND for VBA

' Because of the way excel stores numbers, the + 0.00000001 is to deal with inaccuracy from rounding errors

Function multRound(Amt As Double, factor As Double)

multRound = Round((Amt + 0.0000001) / factor, 0) \* factor

End Function

'Indexation factor

'This function returns the indexation factor to be used in indexation of amounts

' currRate: The rate prior to indexation

' indexSeries: The series used to perform the indexation. The first column must contain the dates, and the second the series

' indexDate: The index series entry which is in the numerator of the index factor

' rounding: The multiple to which the final amount is rounded to

' interval: Frequency of indexation in months

Function indexFctr(indexSeries As String, indexDate As Double, interval As Integer)

Dim dataSeries As Range

Set dataSeries = ThisWorkbook.Names(indexSeries).RefersToRange

Dim datesSeries As Range

Set datesSeries = ThisWorkbook.Names(indexSeries & "Dates").RefersToRange

Dim index As Integer

index = WorksheetFunction.Match(indexDate, datesSeries)

'numerator is the value of the index at the date of indexation

Dim numerator As Double

numerator = dataSeries.Cells(index).value

'get series of index values that are eligible for indexation

'first divide interval by 3 to get to number of quarters, that is frequency which we want to include index values

interval = interval / 3

'Determine starting cell

Dim modulo As Integer

modulo = index Mod interval

'Get set of in scope index values

Dim inScope As Range

Dim started As Integer

started = 0

For I = 1 To index - 1

If (I Mod interval) = modulo Then

If started = 1 Then

Set inScope = Application.Union(inScope, dataSeries.Cells(I))

Else

started = 1

Set inScope = dataSeries.Cells(I)

End If

End If

Next

'denominator is the highest value prior to this

'Dim denominator As Double

denominator = WorksheetFunction.Max(inScope)

'indexation factor is rounded to 3 decimal places

indexFctr = WorksheetFunction.Max(1, Round(numerator / denominator, 3))

End Function

'Indexation function

'This function performs indexation of amounts

' currRate: The rate prior to indexation

' indexSeries: The series used to perform the indexation. The first column must contain the dates, and the second the series

' indexDate: The index series entry which is in the numerator of the index factor

' rounding: The multiple to which the final amount is rounded to

' interval: Frequency of indexation in months

Function IndexAmt(currRate As Double, indexSeries As String, indexDate As Double, rounding As Double, interval As Integer)

'indexation factor recieved using above function

Dim indexFactor As Double

indexFactor = indexFctr(indexSeries, indexDate, interval)

IndexAmt = multRound(currRate \* indexFactor, rounding)

End Function

'indexHELP function

'This function is to perform indexation of the HELP repayment thresholds as per

's154.25 of Higher Education Support Act 2003

'NOTE: A new function may be required to account for change to six monthly

' AWE, once new arrangement become available

'

'Inputs

'rate2005:the base rate to be indexed

'yrStarting:the 1st day of the financial year for which the rate is to apply.

Public Function indexHelp(rate2005 As Double, yrStarting As Date) As Double

'first get the calendar year that we need to pull AWE values for

Dim yrForCalcs As Double

yrForCalcs = year(yrStarting) - 1

'we will now look up AWE for the 4 quarters of this calendar year

'and sum them up as we go

Dim aweSumNumerator As Double

Dim aweSumDenominator As Double

Dim index As Double

Dim datetoFind As Double

'look up March quarter AWE

datetoFind = DateSerial(yrForCalcs, 3, 31)

aweSumNumerator = getEcoPar(datetoFind, "pePTAWE")

aweSumDenominator = getEcoPar(CDbl(DateSerial(2004, 3, 31)), "pePTAWE")

'look up June quarter AWE

datetoFind = DateSerial(yrForCalcs, 6, 30)

aweSumNumerator = aweSumNumerator + getEcoPar(datetoFind, "pePTAWE")

aweSumDenominator = aweSumDenominator + getEcoPar(DateSerial(2004, 6, 30), "pePTAWE")

'look up September quarter AWE

datetoFind = DateSerial(yrForCalcs, 9, 30)

aweSumNumerator = aweSumNumerator + getEcoPar(datetoFind, "pePTAWE")

aweSumDenominator = aweSumDenominator + getEcoPar(DateSerial(2004, 9, 30), "pePTAWE")

'look up December quarter AWE - used to use seasonally adjusted, now does not ;

datetoFind = DateSerial(yrForCalcs, 12, 31)

aweSumNumerator = aweSumNumerator + getEcoPar(datetoFind, "pePTAWE") ' Andrew edit

aweSumDenominator = aweSumDenominator + getEcoPar(DateSerial(2004, 12, 31), "pePTAWE") ' Andrew edit

'get indexed amount

indexHelp = WorksheetFunction.Floor(rate2005 \* aweSumNumerator / aweSumDenominator, 1)

End Function

'getEcoPar Function

'returns the value of an economic parameter on a given date

Public Function getEcoPar(datetoFind As Double, Series As String)

Dim dataSeries As Range

Dim dateSeries As Range

Set dataSeries = ThisWorkbook.Names(Series).RefersToRange

Set dateSeries = ThisWorkbook.Names(Series & "Dates").RefersToRange

Dim index As Integer

index = WorksheetFunction.Match(datetoFind, dateSeries, 1)

getEcoPar = dataSeries.Cells(index)

End Function

' ecoAnnualAverage

' Economic Parameters Annual Average

' Takes an annual average of the four most recent quarters that end at a certain month

' Inputs:

' CurrentPeriod = date of parameter being calculated

' LastQtr = last quarter for series you want to take average of. "MAR", "DEC", "SEP" or "JUN"

' Series = indexation series

Public Function ecoAnnualAverage(CurrentDate As Double, LastQtr As String, Series As String)

Dim index, lookupDate, month, dYear As Double

Dim dataSeries As Range

Dim dateSeries As Range

Set dataSeries = ThisWorkbook.Names(Series).RefersToRange

Set dateSeries = ThisWorkbook.Names(Series & "Dates").RefersToRange

' Get numeric of LastQtr

If LastQtr = "MAR" Then

month = 3

dYear = year(CurrentDate)

ElseIf LastQtr = "DEC" Then

month = 12

dYear = year(CurrentDate) - 1

ElseIf LastQtr = "SEP" Then

month = 9

ydYearear = year(CurrentDate) - 1

ElseIf LastQtr = "JUN" Then

month = 6

dYear = year(CurrentDate) - 1

Else

ecoAnnualAverage = "ERROR-incorrect date format"

End If

' Get the date format of the last quarter we want to look up

lookupDate = DateSerial(dYear, month, 1)

' Find index of relevant date in date series, exact match only

index = WorksheetFunction.Match(lookupDate, dateSeries, 0)

Dim value1, value2, value3, value4 As Double

value1 = WorksheetFunction.index(lookupData, index - 3)

value2 = WorksheetFunction.index(lookupData, index - 2)

value3 = WorksheetFunction.index(lookupData, index - 1)

value4 = WorksheetFunction.index(lookupData, index)

ecoAnnualAverage = (value1 + value2 + value3 + value4) / 4

End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' PAYS() is a generalised vba function for the calculating the number of f/n '

' pension pays between two dates for use with the PTO '

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Public Function PAYS(startDate As Date, EndDate As Date, Optional PensionPay As Boolean = True) As Integer

Dim ReferenceDate As Date

If PensionPay Then

ReferenceDate = 34151

Else

ReferenceDate = 34158

End If

PAYS = Int((ReferenceDate - startDate) / 14) - Int((ReferenceDate - EndDate - 1) / 14)

End Function

'GetPaymentRate function

'Gets the payment rate for a specific quarter of a year starting on a specific date

'Payment is described by mnemoic

Public Function GetPaymentRate(quarter As Integer, yearStarting As Double, mnemonic As String, SheetReference As Range, Dates As Range, mnemRange As Range) As Double

GetPaymentRate = SheetReference.Offset(WorksheetFunction.Match(yearStarting, Dates, 0) - 2 + quarter, WorksheetFunction.Match(mnemonic, mnemRange, 0) - 1)

End Function

'GetPaymentRateOnDate function

'Gets the payment rate of the specified parameter on a specific date

'Inputs:

'startDate: Date at which the payment rate begins to apply

'strMnemonic: The parameter's name in the common parameter spreadsheet

'rngParams: Range containg the parameters of interest

'rngDates: The range of dates on the paramaters sheet

'rngMnems: Range of parameter names to search for the parameter of interest

Public Function GetRateOnDate(startDate As Double, strMnemonic As String, rngParms As Range, rngDates As Range, rngMnems As Range) As Double

'set range to first cell to offset from

Set rngParms = rngParms(1, 1)

GetRateOnDate = rngParms.Offset(WorksheetFunction.Match(startDate, rngDates, 0) - 1, WorksheetFunction.Match(strMnemonic, rngMnems, 0) - 1)

End Function

'FindMe function

'Returns a cell reference to a cell in a range containing a specific value

Public Function FindMe(value As Double, Range As Range) As Range

For Each cell In Range

If cell = value Then

Set FindMe = cell

Exit Function

End If

Next

End Function

'AnuualAverage function

'Returns the annual average of a parameter across the four pay periods

'requires the souce spreadsheet to be in a formay where the four quarterly

'parameters are in four consecutive rows of the same column.

'First two columns of the sheet must contain the start and end dates of the

'period in the format "dd/mm/yyyy"

Public Function AnnualAverage(yearStarting As Double, mnemonic As String, SheetReference As Range, Dates As Range, mnemRange As Range) As Double

'Set Sheet reference to top left hand corner of range\

Set SheetReference = SheetReference(1, 1)

'payment rates for each of the four quarters

Dim Quarter1 As Double

Dim Quarter2 As Double

Dim Quarter3 As Double

Dim Quarter4 As Double

Quarter1 = GetPaymentRate(1, yearStarting, mnemonic, SheetReference, Dates, mnemRange)

Quarter2 = GetPaymentRate(2, yearStarting, mnemonic, SheetReference, Dates, mnemRange)

Quarter3 = GetPaymentRate(3, yearStarting, mnemonic, SheetReference, Dates, mnemRange)

Quarter4 = GetPaymentRate(4, yearStarting, mnemonic, SheetReference, Dates, mnemRange)

'get cell reference of starting date in quarterly worksheet

Dim startCell As Range

Set startCell = FindMe(yearStarting, Dates)

'number of Centrelink payment days in each quarter

Dim Q1Days As Integer

Dim Q2Days As Integer

Dim Q3Days As Integer

Dim Q4Days As Integer

Q1Days = PAYS(startCell.value, startCell.Offset(0, 1).value)

Q2Days = PAYS(startCell.Offset(1, 0).value, startCell.Offset(1, 1).value)

Q3Days = PAYS(startCell.Offset(2, 0).value, startCell.Offset(2, 1).value)

Q4Days = PAYS(startCell.Offset(3, 0).value, startCell.Offset(3, 1).value)

AnnualAverage = (Q1Days \* Quarter1 + Q2Days \* Quarter2 + Q3Days \* Quarter3 + Q4Days \* Quarter4) / (Q1Days + Q2Days + Q3Days + Q4Days)

End Function

'GetIndex function

'Gets the payment rate for a specific quarter of a year starting on a specific date

'Payment is described by mnemoic

Public Function GetIndex(quarter As Integer, yearStarting As Double, mnemonic As String, SheetReference As Range, Dates As Range, mnemRange As Range) As Double

GetIndex = SheetReference.Offset(WorksheetFunction.Match(yearStarting, Dates, 0) - 5 + quarter, WorksheetFunction.Match(mnemonic, mnemRange, 0) - 1)

End Function

'SimpleAverage function

'requires the souce spreadsheet to be in a format where the four quarterly

'parameters are in four consecutive rows of the same column.

'First column of the sheet must contain the start date of the

'period in the format "dd/mm/yyyy"

Public Function SimpleAverage(yearStarting As Double, mnemonic As String, SheetReference As Range, Dates As Range, mnemRange As Range) As Double

'Set Sheet reference to top left hand corner of range\

Set SheetReference = SheetReference(1, 1)

'payment rates for each of the four quarters

Dim Quarter1 As Double

Dim Quarter2 As Double

Dim Quarter3 As Double

Dim Quarter4 As Double

Quarter1 = GetIndex(1, yearStarting, mnemonic, SheetReference, Dates, mnemRange)

Quarter2 = GetIndex(2, yearStarting, mnemonic, SheetReference, Dates, mnemRange)

Quarter3 = GetIndex(3, yearStarting, mnemonic, SheetReference, Dates, mnemRange)

Quarter4 = GetIndex(4, yearStarting, mnemonic, SheetReference, Dates, mnemRange)

'get cell reference of starting date in quarterly worksheet

Dim startCell As Range

Set startCell = FindMe(yearStarting, Dates)

SimpleAverage = (Quarter1 + Quarter2 + Quarter3 + Quarter4) / 4

End Function

Public Function getParm(strWbFile As String, strSheet As String, strName As String, startDate As Double)

Dim strParamRange As String

Dim strNameRange As String

Dim strDateRange As String

strParamRange = "'" & strWbFile & "'!" & strSheet & "Parms"

strNameRange = "'" & strWbFile & "'!" & strSheet & "Names"

strDateRange = "'" & strWbFile & "'!" & strSheet & "Dates"

getParm = GetRateOnDate(startDate, strName, strParamRange, strDateRange, strNameRange)

End Function

## EMTR Output template.xlsb

The following macros are for the EMTR output workbook.

'EMTR chart creator

Sub EMTRcharts()

Dim TmpSheet As Variant

Dim SheetName As String

Dim TemplateSheet As String

Dim KeepArray As Variant

Dim DeleteRange As Range

Dim cell As Range

Dim i As Long

Dim StartTime As Double

Dim SecondsElapsed As Double

'Remember time when macro starts

StartTime = Timer

'Specify name of the template sheet

TemplateSheet = "EMTRchart"

'Set array for EMTR variables

KeepArray = Sheets(TemplateSheet).Range("EMTRLabels").Value2

Application.ScreenUpdating = False

'Temporarily stop Excel updating links

ThisWorkbook.UpdateLinks = xlUpdateLinksNever

'Sheet names and numbers for loops

For Each TmpSheet In ThisWorkbook.Worksheets

SheetName = TmpSheet.Name

'loop through output sheets

If TmpSheet.Index > 4 Then

'Set DeleteRange = Range("A1:CZ1")

'For Each cell In DeleteRange

'If cell.Value <> KeepArray Then cell.EntireColumn.Delete

'Next

'Copy template worksheet and name the same

ThisWorkbook.Sheets(TemplateSheet).Copy After:=ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count)

'Rename

ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count).Name = SheetName & " " & "EMTR"

'Loop over cells, update formula

' First limit the range we are looping over to cells with a formula, to make things run faster

Sheets(SheetName).Select

Range("A1").Select

Range(Selection, Selection.End(xlDown)).Select

Range(Selection, Selection.End(xlToRight)).Select

Selection.Copy

Sheets(SheetName & " " & "EMTR").Select

Range("A1").Select

ActiveSheet.Paste

End If

Next TmpSheet

' Restore defaults

ThisWorkbook.UpdateLinks = xlUpdateLinksAlways

Application.ScreenUpdating = True

'Determine how many seconds code took to run

SecondsElapsed = Round(Timer - StartTime, 2)

'Notify user in seconds

MsgBox "This code ran successfully in " & SecondsElapsed & " seconds", vbInformation

End Sub

'WDR chart creator

Sub WDRcharts()

Dim TmpSheet As Variant

Dim SheetName As String

Dim TemplateSheet As String

Dim StartTime As Double

Dim SecondsElapsed As Double

'Remember time when macro starts

StartTime = Timer

'Specify name of the template sheet

TemplateSheet = "WDRchart"

Application.ScreenUpdating = False

'Temporarily stop Excel updating links

ThisWorkbook.UpdateLinks = xlUpdateLinksNever

'Sheet names and numbers for loops

For Each TmpSheet In ThisWorkbook.Worksheets

SheetName = TmpSheet.Name

'loop through output sheets

If TmpSheet.Index > 4 Then

'Copy template worksheet and name the same

ThisWorkbook.Sheets(TemplateSheet).Copy After:=ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count)

'Rename

ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count).Name = SheetName & " " & "WDR"

'Loop over cells, update formula

' First limit the range we are looping over to cells with a formula, to make things run faster

Sheets(SheetName & " " & "WDR").Select

Range("D1").Select

ActiveCell.Value = SheetName

End If

Next TmpSheet

' Restore defaults

ThisWorkbook.UpdateLinks = xlUpdateLinksAlways

Application.ScreenUpdating = True

'Determine how many seconds code took to run

SecondsElapsed = Round(Timer - StartTime, 2)

'Notify user in seconds

MsgBox "This code ran successfully in " & SecondsElapsed & " seconds", vbInformation

End Sub