

## Summary

This file contains the code of most of the files of this folder. It is just used by me to look for nice code samples. Because it has over 750 pages, it might take a while to open it, especially over the internet.

## File: CompilationOfMacros V2-1

Sub Start\_ReportPreparation()

'Author: Roland Benz, Project Management Excellence

'Date: 26.9.2011

'Input: "ExpectedLayout", "PlsEAC", "ActivityEAC", "ResourceReport"

'Output:

    ""PlsEAC\_FV", "ActivityEAC\_FV", "ResourceReport\_FV"

    ""RD\_MasterDataSet"

'Objective:

    'This Sub contains and starts all procedures necessary to postprocess

    'the input and to prepare the output dataset "RD\_MasterDataSet".

'Main tasks:

    \*\*\*\*

'Duration on my laptop:

'Remarks:

'Program:

\*\*\*\*\*  
\*\*\*\*\*

'0. Preparation (Code in Module2)

\*\*\*\*\*  
\*\*\*\*\*

'Set the workbook path

**Dim** WBName As String, WBPPath As String, **Wb** As **Workbook**

Define Workbook Objekt

WBName = "RD.xlsm"

WBPPath = "C:\Users\t740698\Desktop\"

**Call** z\_OpenAndActivateWb(WBName, WBPPath, Wb)

Call User Defined Function with Arguments

'Create a flat value copy of the input files

'Copy, Paste, Rename "PlsEAC" as "PlsEAC\_FV"

'Copy, Paste, Rename "ActivityEAC" as "ActivityEAC\_FV"

'Copy, Paste, Rename "ResourceReport" as "ResourceReport\_FV"

If 0 Then

**Call** z\_ShNewFlatValueCopy("PlsEAC", "PlsEAC\_FV", "End")

**With** **ActiveWorkbook**.Sheets(Sh\_Source).Tab

Change Workbook Tab Color

**.Color** = RGB(0, 255, 255)

        .TintAndShade = 0

**End With**

**Call** z\_ShNewFlatValueCopy("ActivityEAC", "ActivityEAC\_FV", "End")

**With** **ActiveWorkbook**.Sheets(Sh\_Source).Tab

**.Color** = RGB(0, 255, 255)

        .TintAndShade = 0

**End With**

**Call** z\_ShNewFlatValueCopy("ResourceReport", "ResourceReport\_FV", "End")

```

With ActiveWorkbook.Sheets(Sh_Source).Tab
    .Color = RGB(0, 255, 255)
    .TintAndShade = 0
End With
End If

```

```

'Delete "PIsEAC", "ActivityEAC", "ResourceReport" spreadsheets
If 0 Then
    Call z_ShDelete("PIsEAC", "ActivityEAC", "ResourceReport")
End If

```

```

'Check the column names and positions
If 0 Then '(no longer needed!)
    'Call LayoutCheck
End If

```

```

*****
*****

'1. Copy columns (Code in Module2)
*****
*****

```

```

'Create the new spreadsheet "RD_MasterDataSet"
If 0 Then
    Call z_ShNew("RD_MasterDataSet1", "Begin")
End If

```

```

'Define the column names for the spreadsheet
Dim ShMaster_ColNames As Variant
ShMaster_ColNames = Array( _ Array Constructor
    "PiIdentifier", "ActivityIdentifier", "SyngentaPortfolioLevel1", "SyngentaPortfolioLevel2",
    "PiPortfolioLevel3", "SyngentaPortfolio", "SyngentaProgram", "IsConfidential", "PiStatus",
    "PortfolioType", "PiSubType", "PiTitle", "PiManager", "PiSponsor", "PiResponsibility", "PiLabel",
    "PiStage", "LastGatePassed", "PiScope", "PiCustomer", "PiInvestmentCategory", "PiMarketSegment",
    "PiIndicatorSchedule", "PiIndicatorBudget", "PiIndicatorScope", "PiIndicatorQuality",
    "ReasonForDeviationScope", "ReasonForDeviationSchedule", "ReasonForDeviationQuality", _
    "ReasonForDeviationBudget", "PiLeadAi", "ListOfPiGrouping", "PiListOfActiveIngredients",
    "PiListOfCrops", "PiListOfCropsGroup", "PiListOfRegions", "PiListOfCountries", "PiGeography",
    "PiListOfProductFunctions", "PiPurchaseOrder", "PlanningItemName", "ActivityDescription",
    "ActivityType", "TaskTitle", "ListOfTaskCustomers", "Customer %", "TaskLocation", "TaskStatus",
    "WbsElement", "TaskContact", "ActivityComment", "LegacyTaskIdentifier", "Duration",
    "ExpectedFinishExport", "PlannedStart", "PlannedFinishExport", "ActualStart", "ActualFinishExport",
    "StartNoEarlierThan", _
    "FinishNoLaterThanExport", "AssignedResourcesWithLoad", "AssignedResourcesWithRate",
    "ResourceGroupDescription", "ResourceDescription", "RoleDescription", "TotalNpv", "SalesPeak",
    "BcRequired", "EtcTrialsFullCosts2010", "EtcTrialsFullCosts2011", "EtcTrialsFullCosts2012",
    "EtcTrialsFullCosts2013", "EtcTrialsFullCosts2014", "EtcTrialsFullCosts", "EtcSdFullCosts2010",
    "EtcSdFullCosts2011", "EtcSdFullCosts2012", "EtcSdFullCosts2013", "EtcSdFullCosts2014",
    "EtcSdFullCosts", "EtcOther2010", "EtcOther2011", "EtcOther2012", "EtcOther2013",
    "EtcOther2014", "EtcOther", "EtcFullCosts2010", "EtcFullCosts2011", "EtcFullCosts2012", _
    "EtcFullCosts2013", "EtcFullCosts2014", "EtcFullCosts", "EtcExt2010", "EtcExt2011", "EtcExt2012",
    "EtcExt2013", "EtcExt2014", "EtcExt", "EtcTrials2010", "EtcTrials2011", "EtcTrials2012",
    "EtcTrials2013", "EtcTrials2014", "EtcTrials", "EtcSd2010", "EtcSd2011", "EtcSd2012", "EtcSd2013",

```

```

"EtcSd2014", "EtcSd", "EacTrialsFullCosts2010", "EacTrialsFullCosts2011", "EacTrialsFullCosts2012",
"EacTrialsFullCosts2013", "EacTrialsFullCosts2014", "EacTrialsFullCosts", "EacSdFullCosts2010",
"EacSdFullCosts2011", "EacSdFullCosts2012", _
"EacSdFullCosts2013", "EacSdFullCosts2014", "EacSdFullCosts", "EacOther2010", "EacOther2011",
"EacOther2012", "EacOther2013", "EacOther2014", "EacOther", "EacFullCosts2010",
"EacFullCosts2011", "EacFullCosts2012", "EacFullCosts2013", "EacFullCosts2014", "EacFullCosts",
"EacExt2010", "EacExt2011", "EacExt2012", "EacExt2013", "EacExt2014", "EacExt", "EacTrials2010",
"EacTrials2011", "EacTrials2012", "EacTrials2013", "EacTrials2014", "EacTrials", "EacSd2010",
"EacSd2011", "EacSd2012", _
"EacSd2013", "EacSd2014", "EacSd", "AcTrialsFullCosts2010", "AcTrialsFullCosts2011",
"AcTrialsFullCosts2012", "AcTrialsFullCosts2013", "AcTrialsFullCosts2014", "AcTrialsFullCosts",
"AcSdFullCosts2010", "AcSdFullCosts2011", "AcSdFullCosts2012", "AcSdFullCosts2013",
"AcSdFullCosts2014", "AcSdFullCosts", "AcOther2010", "AcOther2011", "AcOther2012",
"AcOther2013", "AcOther2014", "AcOther", "AcFullCosts2010", "AcFullCosts2011",
"AcFullCosts2012", "AcFullCosts2013", "AcFullCosts2014", "AcFullCosts", "AcExt2010", "AcExt2011",
"AcExt2012", _
"AcExt2013", "AcExt2014", "AcExt", "AcTrials2010", "AcTrials2011", "AcTrials2012", "AcTrials2013",
"AcTrials2014", "AcTrials", "AcSd2010", "AcSd2011", "AcSd2012", "AcSd2013", "AcSd2014", "AcSd")

```

'Add the column names to the spreadsheet

If 0 Then

    Call z\_AddColNames(ShMaster\_ColNames, "RD\_MasterDataSet1", 1, Wb)

End If

```

*****
*****

```

'2. Copy columns (Code in Module3)

```

*****
*****

```

'Create the new spreadsheet "Log\_CopyColActivityEAC"

If 0 Then

    Call z\_ShNew("Log\_CopyColActivityEAC", "Begin")

End If

'Create the new spreadsheet "RD\_MasterDataSet1"

If 0 Then

    Call z\_ShNewFlatValueCopy("RD\_MasterDataSet1", "RD\_MasterDataSet2", "Begin")

End If

'Copy the columns from the spreadsheet "ActivityEAC\_FV"

If 0 Then

    Call z\_ShCopyColumns(ShMaster\_ColNames, "ActivityEAC\_FV", "RD\_MasterDataSet2",  
    "Log\_CopyColActivityEAC")

End If

```

*****
*****

```

'3. Map columns and some format and entry changes (Code in Module3)

```

*****
*****

```

'Create the new spreadsheet "Log\_MapColPIsEAC"

```

If 0 Then
    Call z_ShNew("Log_MapColPisEAC", "Begin")
End If
'Create the new spreadsheet "RD_MasterDataSet"
If 0 Then
    Call z_ShNewFlatValueCopy("RD_MasterDataSet2", "RD_MasterDataSet3", "Begin")
End If

'Define the column names to map (these names must be on both the source and the target
spreadsheets)
Dim ShPisEAC_ColNamesToMap As Variant
ShPisEAC_ColNamesToMap = Array( _
    "PiPortfolioLevel3", "SyngentaProgram", "PiStatus", "PortfolioType", "PiSubType", "PiTitle",
    "PiManager", "PiSponsor", "PiResponsibility", "PiLabel", _
    "PiStage", "LastGatePassed", "PiScope", "PiCustomer", "PiInvestmentCategory", "PiMarketSegment",
    "PiIndicatorSchedule", "PiIndicatorBudget", "PiIndicatorScope", "PiIndicatorQuality", _
    "ReasonForDeviationScope", "ReasonForDeviationSchedule", "ReasonForDeviationQuality",
    "ReasonForDeviationBudget", "PiLeadAi", _
    "ListOfPiGrouping", "PiListOfActiveIngredients", "PiListOfCrops", "PiListOfCropsGroup",
    "PiListOfRegions", "PiListOfCountries", "PiGeography", "PiListOfProductFunctions",
    "PiPurchaseOrder", _
    "TotalNpv", "SalesPeak", "BcRequired")

'Map the columns from the spreadsheet "PisEAC_FV"
If 0 Then
    Call z_ShMapColumns(ShPisEAC_ColNamesToMap, ShMaster_ColNames, "PisEAC_FV",
    "RD_MasterDataSet3", "Log_MapColPisEAC")
End If

'***not found Pis in Logfile_MapPisEAC mapped from online report
Dim SmCExtract1_ColNamesToMap As Variant
SmCExtract1_ColNamesToMap = ShPisEAC_ColNamesToMap
If 0 Then '!!!!!!!!!!!!!!Todo and to test
    Call z_ShMapColumns(SmCExtract1_ColNamesToMap, ShMaster_ColNames, "SmCExtract1",
    "RD_MasterDataSet3", "Log_MapColPisEAC")
End If

'***map in column J PiType (Project/NonProject) from online report (PiType <> PortfolioType)
'change the layout arrays accordingly
If 0 Then '!!!!!!!!!!!!!!Todo and to test
    Dim SmCExtract2_ColNameToMap1 As Variant
    SmCExtract2_ColNameToMap = Array("PiType")
    Call z_ShMapColumns(SmCExtract2_ColNameToMap, ShMaster_ColNames, "SmCExtract2",
    "RD_MasterDataSet3", "Log_MapColSmCExtract")
End If

'***map in column BA with online extract (since not readable format in Infosys Report)
If 0 Then '!!!!!!!!!!!!!!Todo and to test
    Dim SmCExtract3_ColNameToMap1 As Variant
    SmCExtract3_ColNameToMap = Array("Duration")
    Call z_ShMapColumns(SmCExtract3_ColNameToMap, ShMaster_ColNames, "SmCExtract3",
    "RD_MasterDataSet3", "Log_MapColSmCExtract")
End If

```

```

***Change the Format of the date colums BA to BH. Remove for Columns BE to BH with left(,10)
If 0 Then
Dim Date_Array As Variant
Date_Array = Array("Duration", "ExpectedFinishExport", "PlannedStart", "PlannedFinishExport", _
"ActualStart", "ActualFinishExport", "StartNoEarlierThan", "FinishNoLaterThanExport")
    Call z_ChgDateFormat("RD_MasterDataSet3", Date_Array, 0)
Date_Array = Array("ActualStart", "ActualFinishExport", "StartNoEarlierThan",
"FinishNoLaterThanExport")
    Call z_ChgDateFormat("RD_MasterDataSet3", Date_Array, 1)
End If

```

\*\*\*BP Yes/No instead of True/False

```

Dim Val_old As Boolean
Dim Val_new As String

```

Define Boolean and String Variables

```

If 0 Then
    Val_old = False
    Val_new = "No"
    Call z_ChgAttrValue("RD_MasterDataSet3", "BcRequired", Val_old, Val_new)
    Val_old = True
    Val_new = "Yes"
    Call z_ChgAttrValue("RD_MasterDataSet3", "BcRequired", Val_old, Val_new)
End If

```

'Rename the (partly misused SmC) attribute from "PI Purchase Order" to "Grower Need"

```

If 0 Then
    ErrFlg = z_RenameCol("PiPurchaseOrder", "GrowerNeed", "RD_MasterDataSet4")
If ErrFlg = 0 Then
    Stop 'Umbenennung fehlgeschlagen und Name noch nicht existent
End If
End If

```

\*\*\*Write in logfie those with PiStatus on WS and TK level with no entry

```

If 0 Then

```

```

End If

```

```

*****
*****

```

'4. ListOfPiCustomer, find and correct errors, split, cost reduction, rename (Code in Module4)

```

*****
*****

```

'Create the new spreadsheet "Log\_CustomerChk"

```

If 0 Then
    Call z_ShNew("Log_CustomerChk", "Begin")
End If

```

'Create the new spreadsheet "RD\_MasterDataSet"

```

If 0 Then
    Call z_ShNewFlatValueCopy("RD_MasterDataSet3", "RD_MasterDataSet4", "Begin")
End If

```

'Convert numbers in text format to real number format

```

If 0 Then
    Call z_ChgFmt_CostCols("RD_MasterDataSet4", 69, 194)
End If

```

'Write into the logfile a list of all different entries in the ListOfTaskCustomers

```

If 0 Then
    Call z_ListOfSortedAttributeEntries("ListOfTaskCustomers", "RD_MasterDataSet4", 1, 1,
"Log_CustomerChk")
End If

```

'Split the string in the logfile

```

Dim DelimiterList As Variant
Dim ReplacementList As Variant
Dim StringSplit_Dim2 As Variant
Dim RowSize As Long
DelimiterList = Array("(", "%", ",", " ", "")
ReplacementList = Array("@", "@", "@", "@")
RowSize = z_RowSize(2, "Log_CustomerChk")
Sheets("Log_CustomerChk").Cells(1, 3).Value = "z_StringSplit"
If 0 Then
    StringSplit_Dim2 = z_StringSplit(DelimiterList, ReplacementList, 2, RowSize, 2, "Log_CustomerChk",
2, 3, "Log_CustomerChk")
End If

```

'find errors and correct them in the logfile, make a list for PMs to make corrections in SmC

```

If 0 Then
    Call z_ListOfTaskCustomerError1(StringSplit_Dim2, "Log_CustomerChk")
End If

```

```

If 0 Then
    Call z_ListOfTaskCustomerError2(StringSplit_Dim2, "Log_CustomerChk")
End If

```

```

If 0 Then
    Call z_ListOfTaskCustomerError3(StringSplit_Dim2, "Log_CustomerChk")
End If

```

'build the corrected strings in the logfile

```

Dim Rng_From As Range
Dim Rng_To As Range
'To set a range the sheet must be activated!
Sheets("Log_CustomerChk").Select
Set Rng_From = Sheets("Log_CustomerChk").Range(Cells(3, 4), Cells(47, 7))
Set Rng_To = Sheets("Log_CustomerChk").Range(Cells(3, 9), Cells(47, 9))
Sheets("Log_CustomerChk").Cells(1, 9) = "z_BuildSting"
If 0 Then
    Call z_BuildSting(Rng_From, "Log_CustomerChk", Rng_To, "Log_CustomerChk")
End If

```

'replace the corrected strings in the RD\_MasterDataSet

```

Dim Rng_What As Range      Declare a Range Object
Dim Rng_Repl As Range
Dim Rng_To2 As Range

```

'To set a range the sheet must be activated!

Sheets("Log\_CustomerChk").Select

Access Sheet Object: Select it

Set Rng\_What = Sheets("Log\_CustomerChk").Range(Cells(3, 2), Cells(47, 2))

Assign Range to Range Object

Set Rng\_Repl = Sheets("Log\_CustomerChk").Range(Cells(3, 9), Cells(47, 9))

'To set a range the sheet must be activated!

Sheets("RD\_MasterDataSet4").Select

RowSize = z\_RowSize(1, "RD\_MasterDataSet4")

Set Rng\_To2 = Sheets("RD\_MasterDataSet4").Range(Cells(2, 45), Cells(RowSize, 45))

If 0 Then

Call z\_ReplaceWrongAttributEntries("Log\_CustomerChk", Rng\_What, Rng\_Repl,

"RD\_MasterDataSet4", Rng\_To2)

End If

'Split concatenated string in Attribute "ListOfTaskCustomers" and store it in "CustomerName" and "Customer%"

'Reduce all cost fields by the value in the column "Customer%"

If 0 Then

'since a row is copied this should be done at the very end of this macro!

Call z\_Split\_Concatenated\_ListOfTastCustomers("RD\_MasterDataSet4", "ListOfTaskCustomers")

End If

If 0 Then

Call z\_CostReduction\_ListOfTastCustomers("RD\_MasterDataSet4", "Customer %",

"EtcTrialsFullCosts2010")

End If

'Rename the attribute "ListOfTaskCustomers" to "CustomerName"

If 0 Then

ErrFlg = z\_RenameCol("ListOfTaskCustomers", "CustomerName", "RD\_MasterDataSet4")

If ErrFlg = 0 Then

Stop 'Umbenennung fehlgeschlagen und Name noch nicht existent

End If

End If

\*\*\*\*\*

\*\*\*\*\*

'5. Remove whole projects with PiStatus = Evaluation on PiLevel(Code in Module5)

\*\*\*\*\*

\*\*\*\*\*

'Create the new spreadsheet "Log\_RemPiStatus\_Eval"

If 0 Then

Call z\_ShNew("Log\_RemPiStatus\_Eval", "Begin")

End If

'Create the new spreadsheet "RD\_MasterDataSet"

If 0 Then

Call z\_ShNewFlatValueCopy("RD\_MasterDataSet4", "RD\_MasterDataSet5", "Begin")

End If

'Remove those with PiStatus = evaluation

If 0 Then

```

    Call z_RemWholeProject_WithAttrEntryOnPiLev("RD_MasterDataSet5", "Log_RemPiStatus_Eval", _
        "PiIdentifier", "PiStatus", "Evaluation", 0)
End If

'*****
'*****

'6. Remove rows with ActivityIdentifier="MS" (Code in Module5)
'*****
'*****

'Create the new spreadsheet "Log_RemActId_MS"
If 0 Then
    Call z_ShNew("Log_RemActId_MS", "Begin")
End If

'Create the new spreadsheet "RD_MasterDataSet"
If 0 Then
    Call z_ShNewFlatValueCopy("RD_MasterDataSet5", "RD_MasterDataSet6", "Begin")
End If

'Remove those with ActivityIdentifier="MS"
If 0 Then
    Call z_RemRow_WithAttrEntry("RD_MasterDataSet6", "Log_RemActId_MS", _
        "PiIdentifier", "ActivityIdentifier", "MS", 1)
End If

'*****
'*****

'7. Recalculation of EAC Residuals on Pi and WS level(Code in Module6)
'*****
'*****

'Create the new spreadsheet "Log_EACRecalc"
If 0 Then
    Call z_ShNew("Log_EACRecalc", "Begin")
End If

'Create the new spreadsheet "RD_MasterDataSet"
If 0 Then
    Call z_ShNewFlatValueCopy("RD_MasterDataSet6", "RD_MasterDataSet7", "Begin")
End If

'Recalculate the residual EAC costs on the PI and WS levels.
If 0 Then
    Call z_EACRecalc("RD_MasterDataSet7", "Log_EACRecalc")
End If

'Create the new spreadsheet "Log_EACRecalc"
If 0 Then
    Call z_ShNew("Log_ComparisonOnPiLevel", "Begin")
End If

'Check the results of the Macro "z_EACRecalc()"
'by adding the costs from the Sheet Sh_BeforeRecalc

```



```

'by adding the costs from the Sheet Sh_AfterRecalc
If 0 Then
    Call z_ComparisonOnPiLevel("RD_MasterDataSet6", "RD_MasterDataSet7",
"Log_ComparisonOnPiLevel")
End If

If 0 Then
    Call z_ShNew("Log_CostTest", "Begin")
End If
'Test the cost columns (Works only on the defined Layout)
If 0 Then
Dim Row_To As Long
Dim Row_From As Long
Dim NrOfRows As Long
'Do some formatting
NrOfRows = 4
    Call z_CostTestFormating("RD_MasterDataSet7", "Log_CostTest", Row_From, NrOfRows)
'two tests
Row_From = 2
Row_To = 2 + 6 * 0
    Call z_CostsTest("RD_MasterDataSet7", "Log_CostTest", Row_From, Row_To)
Row_From = 3
Row_To = 2 + 6 * 1
    Call z_CostsTest("RD_MasterDataSet7", "Log_CostTest", Row_From, Row_To)
End If

'Compare the calculated costs with those of a SmC extract
If 0 Then
    Call z_CostComparison_GenerateSmCExtract
End If
If 0 Then
    Call z_CostComparisonWithSmCExtract
End If
'*****
'*****

'8. Make the Crop Split(Module7)
'*****
'*****

'Create the new spreadsheet "Log_CropSplit"
If 0 Then
    Call z_ShNew("Log_CropSplit", "Begin")
End If

'Create the new spreadsheet "RD_MasterDataSet"
If 0 Then
    Call z_ShNewFlatValueCopy("RD_MasterDataSet7", "RD_MasterDataSet8", "Begin")
End If
'*** 1. Crop Split
If 0 Then
    Call z_SmCExtractMap_CropProtectionToCropSplit("NEW_2011_Pi Split by Crop",
"RD_MasterDataSet8", "Log_CropSplit")
End If

```

```

*****
*****

'9. Make checks on the data (Code in Module8)
*****
*****

'Create the new spreadsheet "Log_DataCks"
If 1 Then
    Call z_ShNew("Log_DataCks", "Begin")
End If

'Create the new spreadsheet "RD_MasterDataSet"
If 1 Then
    Call z_ShNewFlatValueCopy("RD_MasterDataSet8", "RD_MasterDataSet9", "Begin")
End If
*** 1. Check whether the values of the attributes make sense
If 0 Then
    Call DataCheckForErrorsAndViolatedPreconditions
End If

*****
*****

'10. Auxiliary function are in Module 9 (Code in Module9)
*****
*****

End Sub

```

```

Public Function z_OpenAndActivateWb(WBName As String, WBPath As String, ByRef Wb As
Workbook)

```

Function Definition:  
Keywords: Public, ByRef, As

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Objective: Open the Workbook RD.xlsb if not already open and activate it.

'look if wbName is existent in workbooks list

Dim i As Long

For i = Workbooks.Count To 1 Step -1

Loop: Collection of all open Workbooks

If Workbooks(i).Name = WBName Then Exit For

Access one Workbook of Collection

Next

'if wbName is existent in workbooks list, then i<>0-> activate workbook

'if wbName is not existent then i=0-> open workbook, activate workbook

If i <> 0 Then

Set Wb = GetObject(WBPath & WBName)

If Condition Then

Wb.Activate

Else

Else

Else If

Set Wb = Workbooks.Open(WBPath & WBName)

End If

Wb.Activate

End If

End Function

Sub z\_ShNewFlatValueCopy(Sh As String, Sh\_new As String, Optional Where As String, Optional ByRef Sh\_Ref As Worksheet)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Name of the Sh to copy, Name of the new Sh\_new, where to place the new Sh\_new,

```

' before or after some Sh_Ref, at the begin or the end
Call z_ShAdd(Where, Sh_Ref)
On Error Resume Next
ActiveSheet.Name = Sh_new
If Err.Number <> 0 Then
    Application.DisplayAlerts = False
    ActiveSheet.Delete
    Application.DisplayAlerts = True
    Sheets(Sh_new).Cells.ClearContents
End If
On Error GoTo 0
Sheets(Sh).Cells.Copy
Sheets(Sh_new).Range("A1").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
Sheets(Sh_new).Columns("A:GA").ColumnWidth = 20
End Sub

```

Copy / Paste whole sheet content into new sheet:  
> Copy Sh  
> Select Sh\_new  
> Paste Sh into Sh\_new

Access column: change column width

```

Sub z_ShDelete(Sh1 As String, Sh2 As String, Sh3 As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: "PlsEAC", "ActivityEAC", "ResourceReport"
Application.DisplayAlerts = False
Sheets(Sh1).Delete
Sheets(Sh2).Delete
Sheets(Sh3).Delete
Application.DisplayAlerts = True
End Sub

```

Delete Sheet

```

Function z_ShNew(Sh As String, Optional Where As String, Optional ByRef Sh_Ref As Worksheet,
Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Name of the new Sh, where to place the new Sh, before or after some Sh_Ref, at the begin or
the end
On Error Resume Next
WorksheetExists = (Sheets(Sh).Name <> "")
On Error GoTo 0
If WorksheetExists = False Then
Call z_ShAdd(Where, Sh_Ref)
ActiveSheet.Name = Sh 'Worksheets.Add(Before:=Worksheets(1)).Name = Sh
End If
'Clear contents
Sheets(Sh).Activate
ActiveSheet.Cells.Select
Selection.ClearContents
'Format
Sheets(Sh).Columns.ColumnWidth = 20
Sheets(Sh).Rows.RowHeight = 15
End Function

```

Boolean Test

Boolean Test Keywords: True / False

```

Function z_ShAdd(Optional Where As String, Optional ByRef Sh_Ref As Worksheet)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

'Date: 26.9.2011

'Input: where to place the added Sh, before or after some Sh\_Ref, at the begin or the end

**If Not Sh\_Ref Is Nothing Then**

Boolean Test Keyword for Objects: Nothing

    If Where = ("Before:") Then

**Sheets.Add Sh\_Ref**

    Elseif Where = ("After:") Then

**Sheets.Add , Sh\_Ref**

    Else

**Sheets.Add Before:=Sheets(1)**

    End If

Else

Add a new sheet at different positions

    If Where = ("End") Then

**Sheets.Add After:=Sheets(Sheets.Count)**

    Elseif Where = ("Begin") Then

        Sheets.Add Before:=Sheets(1)

    Else

        Sheets.Add Before:=Sheets(1)

    End If

End If

End Function

Function z\_AddColNames(ByRef ColNames As Variant, Optional Sh As String, Optional Row As Integer

**= 1**, Optional ByRef Wb As Workbook)

Default Value for Arguments

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Size\_ColNames = **UBound(ColNames)**

size for arrays: upper bound

**For j = 0 To Size\_ColNames**

Loop: For ... Next

**Cells(Row, j + 1) = ColNames(j)**

Cell Object: Assign Value from Array Object

**Next j**

'Selection.EntireColumn.AutoFit

End Function

Sub LayoutCheck(Sh As String)

'Author: Roland Benz, Project Management Excellence

'Date: 19.9.2011

'Input: "ActivityEAC\_FV", "PlsEAC\_FV", "ResourceReport\_FV"

'Output: Filled file "Logfile\_LayoutCheck"

'Objective: To check whether the input files have the defined structure (column names and position)

'Main tasks:

    \*\*\*1. Check the spreadsheet "ActivityEAC\_FV"

    \*\*\*2. Check the spreadsheet "PlsEAC\_FV"

    \*\*\*3. Check the spreadsheet "ResourceReport\_FV"

'Duration on my laptop:

'Remarks:

'Program:

'Create the new spreadsheet "Logfile\_LayoutCheck"

```

If 1 Then 'flag: 1=on, 0=off
On Error Resume Next
WorksheetExists1 = (Sheets("Logfile_LayoutCheck").Name <> "")
On Error GoTo 0
If WorksheetExists1 = False Then
    Worksheets.Add(Before:=Worksheets(1)).Name = "Logfile_LayoutCheck"
End If
'Logfile clear contents
Sheets("Logfile_LayoutCheck").Activate
ActiveSheet.Cells.Select           Select and clear all content in a sheet
Selection.ClearContents
'logfile add column names
Sheets("Logfile_LayoutCheck").Activate
Dim ilog As Integer
ilog = 1
Cells(ilog, 2) = "ColumnNameExp": Cells(ilog, 3) = "ColumnExp": Cells(ilog, 4) = "Flag"
ilog = ilog + 1
Selection.EntireColumn.AutoFit     change column width
End If 'flag: 1=on, 0=off

```

```

**** 1. Check the spreadsheet "ActivityEAC_FV"
Sheets("ActivityEAC_FV").Activate

```

```

'List of defined columns (expected column names and positions)
Dim LayoutExp As Variant

```

```

LayoutExp = Array( _
    "PiIdentifier", A_, "IsConfidential", B_, "ProjectFilter", C_, "Task_list", D_, "ActivityIdentifier", E_,
    "SyngentaPortfolioLevel1", F_, "SyngentaPortfolioLevel2", G_, "SyngentaPortfolio", H_,
    "PlanningItemName", I_, "ActivityDescription", J_, "ActivityType", K_, "TaskTitle", L_, "Duration", M_,
    "ExpectedFinishExport", N_, "PlannedStart", O_, _
    "PlannedFinishExport", P_, "ActualStart", Q_, "ActualFinishExport", R_, "StartNoEarlierThan", S_,
    "FinishNoLaterThanExport", T_, "ListOfTaskCustomers", U_, "ListOfTaskCustomers", U_,
    "AssignedResourcesWithLoad", V_, "AssignedResourcesWithRate", W_, "ResourceGroupDescription",
    W_, "ResourceDescription", W_, "RoleDescription", W_, "EtcTrialsFullCosts2014", X_,
    "EtcTrialsFullCosts2013", Y_, "EtcTrialsFullCosts2012", Z_, _
    "EtcTrialsFullCosts2011", AA_, "EtcTrialsFullCosts2010", AB_, "EtcTrialsFullCosts", AC_,
    "EtcSdFullCosts2014", AD_, "EtcSdFullCosts2013", AE_, "EtcSdFullCosts2012", AF_,
    "EtcSdFullCosts2011", AG_, "EtcSdFullCosts2010", AH_, "EtcSdFullCosts", AI_, "EtcOther2014", AJ_,
    "EtcOther2013", AK_, "EtcOther2012", AL_, "EtcOther2011", AM_, "EtcOther2010", AN_, "EtcOther",
    AO_, _
    "EtcFullCosts2014", AP_, "EtcFullCosts2013", AQ_, "EtcFullCosts2012", AR_, "EtcFullCosts2011", AS_,
    "EtcFullCosts2010", AT_, "EtcFullCosts", AU_, "EtcExt2014", AV_, "EtcExt2013", AW_, "EtcExt2012",
    AX_, "EtcExt2011", AY_, "EtcExt2010", AZ_, "EtcExt", BA_, "EtcTrials2014", BB_, "EtcTrials2013", BC_,
    "EtcTrials2012", BD_, _
    "EtcTrials2011", BE_, "EtcTrials2010", BF_, "EtcTrials", BG_, "EtcSd2014", BH_, "EtcSd2013", BI_,
    "EtcSd2012", BJ_, "EtcSd2011", BK_, "EtcSd2010", BL_, "EtcSd", BM_, "EacTrialsFullCosts2014", BN_,
    "EacTrialsFullCosts2013", BO_, "EacTrialsFullCosts2012", BP_, "EacTrialsFullCosts2011", BQ_,
    "EacTrialsFullCosts2010", BR_, "EacTrialsFullCosts", BS_, _
    "EacSdFullCosts2014", BT_, "EacSdFullCosts2013", BU_, "EacSdFullCosts2012", BV_,
    "EacSdFullCosts2011", BW_, "EacSdFullCosts2010", BX_, "EacSdFullCosts", BY_, "EacOther2014", BZ_,
    "EacOther2013", CA_, "EacOther2012", CB_, "EacOther2011", CC_, "EacOther", CD_,
    "EacOther2010", CE_, "EacFullCosts2014", CF_, "EacFullCosts2013", CG_, "EacFullCosts2011", CH_, _

```

```

"EacFullCosts2012", CI_, "EacFullCosts2010", CJ_, "EacFullCosts", CK_, "EacExt2014", CL_,
"EacExt2013", CM_, "EacExt2012", CN_, "EacExt2011", CO_, "EacExt2010", CP_, "EacExt", CQ_,
"EacTrials2014", CR_, "EacTrials2013", CS_, "EacTrials2012", CT_, "EacTrials2011", CU_,
"EacTrials2010", CV_, "EacTrials", CW_, _
"EacSd2014", CX_, "EacSd2013", CY_, "EacSd2012", CZ_, "EacSd2011", DA_, "EacSd2010", DB_,
"EacSd", DC_, "AcTrialsFullCosts2014", DD_, "AcTrialsFullCosts2013", DE_, "AcTrialsFullCosts2012",
DF_, "AcTrialsFullCosts2010", DG_, "AcTrialsFullCosts2011", DH_, "AcTrialsFullCosts", DI_,
"AcSdFullCosts2014", DJ_, "AcSdFullCosts2013", DK_, "AcSdFullCosts2012", DL_, _
"AcSdFullCosts2011", DM_, "AcSdFullCosts2010", DN_, "AcSdFullCosts", DO_, "AcOther2014", DP_,
"AcOther2013", DQ_, "AcOther2012", DR_, "AcOther2011", DS_, "AcOther2010", DT_, "AcOther",
DU_, "AcFullCosts2014", DV_, "AcFullCosts2013", DW_, "AcFullCosts2012", DX_, "AcFullCosts2011",
DY_, "AcFullCosts2010", DZ_, "AcFullCosts", EA_, _
"AcExt2014", EB_, "AcExt2013", EC_, "AcExt2012", ED_, "AcExt2011", EE_, "AcExt2010", EF_, "AcExt",
EG_, "AcTrials2014", EH_, "AcTrials2013", EI_, "AcTrials2012", EJ_, "AcTrials2011", EK_,
"AcTrials2010", EL_, "AcTrials", EM_, "AcSd2014", EN_, "AcSd2013", EO_, "AcSd2012", EP_, _
"AcSd2011", EQ_, "AcSd2010", ER_, "AcSd", ES_, "TaskLocation", ET_, "TaskStatus", EU_)

```

'Find Column numbers of Column names

If 1 Then 'flag: 1=on, 0=off

'Actual

Dim CellIndexStrAct As String

**Dim CellIndexArrAct() As String** Declare String Array

Dim ColIndexAct As String

'Expected

Dim ColNameExp\_i As String

Dim ColIndexExp\_i As String

**Dim EnumColIndexExp\_i As Enum\_Col** Declare Enum\_Col Object

'iterate through the expected values of the array

**For i = LBound(LayoutExp) To UBound(LayoutExp) Step 2** Array: lower bound and upper bound

'read from the array

ColNameExp\_i = LayoutExp(i)

ColIndexExp\_i = LayoutExp(i + 1)

EnumColIndexExp\_i = ColIndexExp\_i

'find column name

**On Error GoTo NameExpectedNotExist:** Jump to label with goto

Sheets("ActivityEAC\_FV").Activate

**Rows("1:1").Find(What:=CStr(ColNameExp\_i), LookAt:=xlWhole).Select** find string in whole spreadsheet row

'find column index

**CellIndexStrAct = ActiveCell.Address(ReferenceStyle:=xlR1C1)**

**CellIndexArrAct = Split(CellIndexStrAct, "C")** way to find column index of active cell

**ColIndexAct = CInt(CellIndexArrAct(1))** there is probably a better way

'write out into the logfile

Sheets("Logfile\_LayoutCheck").Cells(iLog, 1) = "ActivityEAC\_FV"

Sheets("Logfile\_LayoutCheck").Cells(iLog, 2) = ColNameExp\_i

Sheets("Logfile\_LayoutCheck").Cells(iLog, 3) = ColIndexExp\_i

If EnumColIndexExp\_i = CInt(ColIndexAct) Then

    Sheets("Logfile\_LayoutCheck").Cells(iLog, 4) = "OK: attribute name and position as expected"

Elseif EnumColIndexExp\_i = 0 Then

    Sheets("Logfile\_LayoutCheck").Cells(iLog, 4) = "wrong: attribute name not as expected"

Else

    Sheets("Logfile\_LayoutCheck").Cells(iLog, 4) = "wrong: attribute position not as expected"

End If

```
    ilog = ilog + 1
Next i
End If 'flag: 1=on, 0=off
```

```
***2. Check the spreadsheet "PIsEAC_FV"
Sheets("PIsEAC_FV").Activate
```

```
'List of defined columns (expected column names and positions)
```

```
LayoutExp = Array( _
"PIPortfolioLevel1", A_, "PIPortfolioLevel2", B_, "PIPortfolioLevel3", C_, "SyngentaPortfolio", E_,
"SyngentaProgram", D_, "PiIdentifier", F_, "PiStatus", G_, "PortfolioType", H_, "IsConfidential", I_,
"ProjectFilter", J_, "PiSubType", K_, "PiTitle", L_, "PiResponsibility", M_, "PiLabel", N_, "PiStage", O_,
_
"LastGatePassed", P_, "PiManager", Q_, "PiSponsor", R_, "PiScope", S_, "StartDate", T_,
"PiCustomer", U_, "PiEndDate", V_, "PiInvestmentCategory", W_, "PiMarketSegment", X_,
"PiIndicatorSchedule", Y_, "PiIndicatorBudget", Z_, "PiIndicatorScope", AA_, "PiIndicatorQuality",
AB_, "ReasonForDeviationScope", AC_, "ReasonForDeviationSchedule", AD_, _
"ReasonForDeviationQuality", AE_, "ReasonForDeviationBudget", AF_, "PiLeadAi", AG_,
"PiPurchaseOrder", AH_, "ListOfPiGrouping", AI_, "PiListOfActiveIngredients", AJ_, "PiListOfCrops",
AK_, "PiListOfCropsGroup", AL_, "PiListOfRegions", AM_, "PiListOfCountries", AN_, "PiGeography",
AO_, "PiListOfProductFunctions", AP_, "EtcTrialsFullCosts2014", AQ_, "EtcTrialsFullCosts2013", AR_,
"EtcTrialsFullCosts2012", AS_, _
"EtcTrialsFullCosts2011", AT_, "EtcTrialsFullCosts2010", AU_, "EtcTrialsFullCosts", AV_,
"EtcSdFullCosts2014", AW_, "EtcSdFullCosts2013", AX_, "EtcSdFullCosts2011", AY_,
"EtcSdFullCosts2012", AZ_, "EtcSdFullCosts2010", BA_, "EtcSdFullCosts", BB_, "EtcOther2013", BC_,
"EtcOther2014", BD_, "EtcOther2012", BE_, "EtcOther2011", BF_, "EtcOther2010", BG_, "EtcOther",
BH_, _
"EtcFullCosts2014", BI_, "EtcFullCosts2013", BJ_, "EtcFullCosts2012", BK_, "EtcFullCosts2011", BL_,
"EtcFullCosts2010", BM_, "EtcFullCosts", BN_, "EtcExt2014", BO_, "EtcExt2013", BP_, "EtcExt2012",
BQ_, "EtcExt2011", BR_, "EtcExt2010", BS_, "EtcExt", BT_, "EtcTrials2014", BU_, "EtcTrials2013", BV_,
"EtcTrials2012", BW_, _
"EtcTrials2011", BX_, "EtcTrials2010", BY_, "EtcTrials", BZ_, "EtcSd2014", CA_, "EtcSd2012", CB_,
"EtcSd2013", CC_, "EtcSd2011", CD_, "EtcSd2010", CE_, "EtcSd", CF_, "EacTrialsFullCosts2014", CG_,
"EacTrialsFullCosts2013", CH_, "EacTrialsFullCosts2012", CI_, "EacTrialsFullCosts2011", CJ_,
"EacTrialsFullCosts2010", CK_, "EacTrialsFullCosts", CL_, _
"EacSdFullCosts2014", CM_, "EacSdFullCosts2012", CN_, "EacSdFullCosts2013", CO_,
"EacSdFullCosts2011", CP_, "EacSdFullCosts2010", CQ_, "EacSdFullCosts", CR_, "EacOther2014", CS_,
"EacOther2013", CT_, "EacOther2012", CU_, "EacOther2011", CV_, "EacOther2010", CW_,
"EacOther", CX_, "EacFullCosts2014", CY_, "EacFullCosts2013", CZ_, "EacFullCosts2012", DA_, _
"EacFullCosts2011", DB_, "EacFullCosts2010", DC_, "EacFullCosts", DD_, "EacExt2014", DE_,
"EacExt2013", DF_, "EacExt2012", DG_, "EacExt2011", DH_, "EacExt2010", DI_, "EacExt", DJ_,
"EacTrials2014", DK_, "EacTrials2013", DL_, "EacTrials2012", DM_, "EacTrials2011", DN_,
"EacTrials2010", DO_, "EacTrials", DP_, _
"EacSd2014", DQ_, "EacSd2013", DR_, "EacSd2012", DS_, "EacSd2011", DT_, "EacSd2010", DU_,
"EacSd", DV_, "AcTrialsFullCosts2014", DW_, "AcTrialsFullCosts2013", DX_, "AcTrialsFullCosts2012",
DY_, "AcTrialsFullCosts2011", DZ_, "AcTrialsFullCosts2010", EA_, "AcTrialsFullCosts", EB_,
"AcSdFullCosts2014", EC_, "AcSdFullCosts2013", ED_, "AcSdFullCosts2012", EE_, _
"AcSdFullCosts2011", EF_, "AcSdFullCosts2010", EG_, "AcSdFullCosts", EH_, "AcOther2014", EI_,
"AcOther2013", EJ_, "AcOther2011", EK_, "AcOther2012", EL_, "AcOther2010", EM_, "AcOther", EN_,
"AcFullCosts2013", EO_, "AcFullCosts2014", EP_, "AcFullCosts2011", EQ_, "AcFullCosts2012", ER_,
"AcFullCosts2010", ES_, "AcFullCosts", ET_, _
```

```
"AcExt2014", EU_, "AcExt2013", EV_, "AcExt2011", EW_, "AcExt2012", EX_, "AcExt2010", EY_,
"AcExt", EZ_, "AcTrials2014", FA_, "AcTrials2013", FB_, "AcTrials2012", FC_, "AcTrials2011", FD_,
"AcTrials2010", FE_, "AcTrials", FF_, "AcSd2014", FG_, "AcSd2013", FH_, "AcSd2012", FI_, _
"AcSd2011", FJ_, "AcSd2010", FK_, "AcSd", FL_, "PIdentifier2", FM_)
```

```
'Find Column numbers of Column names
```

```
If 1 Then 'flag: 1=on, 0=off
```

```
'iterate through the expected values of the array
```

```
For i = LBound(LayoutExp) To UBound(LayoutExp) Step 2
```

```
    'read from the array
```

```
    ColNameExp_i = LayoutExp(i)
```

```
    ColIndexExp_i = LayoutExp(i + 1)
```

```
    EnumColIndexExp_i = ColIndexExp_i
```

```
    'find column name
```

```
    On Error GoTo NameExpectedNotExist:
```

```
    Sheets("PlsEAC_FV").Activate
```

```
    Rows("1:1").Find(What:=CStr(ColNameExp_i), LookAt:=xlWhole).Select
```

```
    'find column index
```

```
    CellIndexStrAct = ActiveCell.Address(ReferenceStyle:=xlR1C1)
```

```
    CellIndexArrAct = Split(CellIndexStrAct, "C")
```

```
    ColIndexAct = CInt(CellIndexArrAct(1))
```

```
    'write out into the logfile
```

```
    Sheets("Logfile_LayoutCheck").Cells(iLog, 1) = "PlsEAC_FV"
```

```
    Sheets("Logfile_LayoutCheck").Cells(iLog, 2) = ColNameExp_i
```

```
    Sheets("Logfile_LayoutCheck").Cells(iLog, 3) = ColIndexExp_i
```

```
    If EnumColIndexExp_i = CInt(ColIndexAct) Then
```

```
        Sheets("Logfile_LayoutCheck").Cells(iLog, 4) = "OK: attribute name and position as expected"
```

```
    ElseIf EnumColIndexExp_i = 0 Then
```

```
        Sheets("Logfile_LayoutCheck").Cells(iLog, 4) = "wrong: attribute name not as expected"
```

```
    Else
```

```
        Sheets("Logfile_LayoutCheck").Cells(iLog, 4) = "wrong: attribute position not as expected"
```

```
    End If
```

```
    iLog = iLog + 1
```

```
Next i
```

```
End If 'flag: 1=on, 0=off
```

```
***3. Check the spreadsheet "ResourceReport_FV"
```

```
Sheets("ResourceReport_FV").Activate
```

```
'List of defined columns (expected column names and positions)
```

```
LayoutExp = Array( _
```

```
"PIIdentifier", A_, "PIPortfolioLevel1", B_, "PIPortfolioLevel2", C_, "PIPortfolioLevel3", D_,
```

```
"PIPortfolioLevel4", E_, "PIPortfolioLevel5", F_, "TaskIdentifier", G_, "ResourceIdentifier", H_, "Year",
```

```
I_, "ResourceGroupDescription", J_, "ResourceDescription", K_, "RoleDescription", L_, "CostUnit",
```

```
M_, "PlannedAmountOfCostUnit", N_)
```

```
'Find Column numbers of Column names
```

```
If 1 Then 'flag: 1=on, 0=off
```

```
'iterate through the expected values of the array
```

```
For i = LBound(LayoutExp) To UBound(LayoutExp) Step 2
```

```
    'read from the array
```

```
    ColNameExp_i = LayoutExp(i)
```

```
    ColIndexExp_i = LayoutExp(i + 1)
```

```
    EnumColIndexExp_i = ColIndexExp_i
```



```

'find column name
On Error GoTo NameExpectedNotExistent:
Sheets("ResourceReport_FV").Activate
Rows("1:1").Find(What:=CStr(ColNameExp_i), LookAt:=xlWhole).Select
'find column index
CellIndexStrAct = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexArrAct = Split(CellIndexStrAct, "C")
ColIndexAct = CInt(CellIndexArrAct(1))
'write out into the logfile
Sheets("Logfile_LayoutCheck").Cells(ilog, 1) = "ResourceReport_FV"
Sheets("Logfile_LayoutCheck").Cells(ilog, 2) = ColNameExp_i
Sheets("Logfile_LayoutCheck").Cells(ilog, 3) = ColIndexExp_i
If EnumColIndexExp_i = CInt(ColIndexAct) Then
    Sheets("Logfile_LayoutCheck").Cells(ilog, 4) = "OK: attribute name and position as expected"
Elseif EnumColIndexExp_i = 0 Then
    Sheets("Logfile_LayoutCheck").Cells(ilog, 4) = "wrong: attribute name not as expected"
Else
    Sheets("Logfile_LayoutCheck").Cells(ilog, 4) = "wrong: attribute position not as expected"
End If
ilog = ilog + 1
Next i
End If 'flag: 1=on, 0=off

```

```

'On error goto
NameExpectedNotExistent:
EnumColIndexExp_i = 0
Resume Next

```

```

'Activate the logfile and the filter
Sheets("Logfile_LayoutCheck").Select
RowSize = If(IsEmpty(Range("B1048576")), Range("B1048576").End(xlUp).Row, 1048576)
ColSize = 10
If ActiveSheet.AutoFilterMode = False Then
    Rows("1:1").AutoFilter 'Data>Filter
End If
'Filter the wrong entries
ActiveSheet.Range(Cells(2, 1), Cells(RowSize, ColSize)).AutoFilter Field:=4, Criteria1:="wrong*"
End Sub

```

```

Function z_ShCopyColumns(ByRef Arr_ColNames As Variant, Sh As String, Sh_new As String, Sh_log
As String)

```

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

'iterate through the ColNames array

```

```

Dim ColName_i As String

```

```

For i = LBound(Arr_ColNames) To UBound(Arr_ColNames) Step 1

```

```

    'find column index

```

```

    ColName_i = CStr(Arr_ColNames(i))

```

```

    Sh_ColIndex_i = z_GetColumnIndex(ColName_i, 1, Sh)

```

```

    If Sh_ColIndex_i <> 0 Then

```

```

        'find row size

```

```

        RowSize = z_RowSize(1, Sh)

```

```

        'copy/paste

```

```

    Sheets(Sh).Range(Cells(1, Sh_ColIndex_i), Cells(RowSize, Sh_ColIndex_i)).Copy
Destination:=Sheets(Sh_new).Cells(1, i + 1)
    Sheets(Sh_new).Cells(1, i + 1).Font.Color = RGB(0, 0, 255)
Else
    'write into the Sh_new
    Sheets(Sh_new).Cells(1, i + 1).Font.Color = RGB(255, 0, 0)
    'write into the Sh_log
    ilog = ilog + 1
    Sheets(Sh_log).Cells(ilog, 2) = ColName_i
    Sheets(Sh_log).Cells(ilog, 3) = "not found": ilog = ilog + 1
End If
Next i
'column width
Sheets(Sh_new).Activate
ActiveSheet.Cells.Select
Selection.ColumnWidth = 20
End Function

```

Function z\_ShMapColumns(ByRef Arr\_ColNames As Variant, ByRef Arr\_ColNames\_new As Variant, Sh As String, Sh\_new As String, Sh\_log As String, Optional ByRef Wb As Workbook)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Application.ScreenUpdating = False

Dim Start As Date: Dim Duration As Date

[Date object to measure time](#)

Start = Now() 'to measure the duration of the code

[Now\(\) Time now](#)

Dim ilog As Integer

ilog = 2

'Determine the column indizes in Sh and Sh\_new,

Dim ColName\_i As String

ReDim ColIndices(0 To UBound(Arr\_ColNames), 0 To 2) As Variant

[ReDim: Give Array more Memory](#)

For i = LBound(Arr\_ColNames) To UBound(Arr\_ColNames) Step 1

ColName\_i = CStr(Arr\_ColNames(i))

ColIndices(i, 0) = ColName\_i

ColIndices(i, 1) = z\_GetColumnIndex(ColName\_i, 1, Sh)

ColIndices(i, 2) = z\_GetColumnIndex(ColName\_i, 1, Sh\_new)

Debug.Print ColIndices(i, 0)

Debug.Print ColIndices(i, 1)

Debug.Print ColIndices(i, 2)

If ColIndices(i, 1) = 0 Then

'write errors into the logfile

Sheets(Sh\_log).Cells(ilog, 2) = "ColNames"

Sheets(Sh\_log).Cells(ilog, 3) = i

Sheets(Sh\_log).Cells(ilog, 4) = "not found in Sh"

ilog = ilog + 1

Stop 'make sure you find all!!!!!!

Elseif ColIndices(i, 2) = 0 Then

'write errors into the logfile

Sheets(Sh\_log).Cells(ilog, 2) = "ColNames"

Sheets(Sh\_log).Cells(ilog, 3) = i

Sheets(Sh\_log).Cells(ilog, 4) = "not found in Sh\_new"

ilog = ilog + 1

Stop 'make sure you find all!!!!!!

```

End If
Next i

'Determine the row size in Sh_new
Sheets(Sh_new).Activate
RowSize_new = z_RowSize(1, Sh_new)

'Check whether column F is the "PIdentifier" and "Sh_new"
If PIdentifierCheck(Sh, 1, 6) = False Then
    Stop
End If
'Check whether column A is the "PIdentifier" and select Column A "PIdentifier" in "Sh_new"
If PIdentifierCheck(Sh_new, 1, 1) = False Then
    Stop
Else
    Range(Cells(2, 1), Cells(RowSize_new, 1)).Select    Define Range with two cell objects
End If

'Iterate through the rows with "rcheck" = PIdentifier
For Each rcheck In Selection.Cells    Cell object collection
    'if rcheck is found in Sh then perform the mapping
    If Not Sheets(Sh).Columns("F:F").Find(What:=rcheck, LookAt:=xlWhole) Is Nothing Then    check if cell object is not empty
        'iterate through the columns
        For j = LBound(ColIndices) To UBound(ColIndices) Step 1
            'read the indices
            ColIndex_j = ColIndices(j, 1)
            ColIndex_new_j = ColIndices(j, 2)
            'map
            rcheck.Offset(0, (ColIndex_new_j) - 1).Value = _
                Sheets(Sh).Columns("F:F").Find(What:=rcheck, LookAt:=xlWhole).Offset(0, (ColIndex_j) -
6)
        Next j
    Else
        'write errors into the logfile
        Sheets(Sh_log).Cells(iLog, 2) = rcheck 'writes out PIdentifier
        Sheets(Sh_log).Cells(iLog, 4) = "not found"
    End If
Next
Sheets(Sh_new).Rows.RowHeight = 15
'Write the durations into the logfile
Duration = Now() - Start
Sheets(Sh_log).Cells(iLog + 2, 1) = "Duration" & CStr(Duration): iLog = iLog + 1    & concat
                                         CStr convert to string

Application.ScreenUpdating = True
Exit Function
NameExpectedNotExist:    Label to jump on error
    'write into the RD_MasterDataSet_1
    Sheets("RD_MasterDataSet_1").Cells(1, EnumColIndexExp_i) = ColNameExp_i
    Sheets("RD_MasterDataSet_1").Cells(1, EnumColIndexExp_i).Font.Color = RGB(255, 255, 0)
    'write into the logfile
    iLog = iLog + 1
    Sheets("Logfile_MapPIsEAC").Cells(iLog, 2) = ColNameExp_i
    Sheets("Logfile_MapPIsEAC").Cells(iLog, 3) = "not found": iLog = iLog + 1

```

```

    NotfoundFlag = 0
    Resume Next
End Function

```

```

Function PidentifierCheck(Sh As String, Sh_row As Long, Sh_col As Long) As Boolean
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
If Sheets(Sh).Cells(Sh_row, Sh_col) = "Pidentifier" Then
    PidentifierCheck = True
Else
    PidentifierCheck = False
End If
End Function

```

```

Function z_ChgAttrValue(Sh As String, Attr As String, Val_old As Variant, Val_new As Variant)
Sheets(Sh).Activate
Col_Attr = z_GetColumnIndex(Attr, 1, Sh)
RowSize = z_RowSize(1, Sh)
Dim Attr_Value As Variant
For Row = 2 To RowSize
    Attr_Value = Cells(Row, Col_Attr)
    If Attr_Value = Val_old Then
        Cells(Row, Col_Attr).Value = Val_new
    End If
    If (Row Mod 100) = 0 Then
        Debug.Print Row
    End If
Next Row
End Function

```

```

Function z_ChgDateFormat(Sh As String, ByRef Attr As Variant, Optional Fkt As Integer)
Sheets(Sh).Activate
RowSize = z_RowSize(1, Sh)
ReDim Col_Attr(0 To UBound(Attr)) As Variant
Dim SearchString As String
For iAttr = 0 To UBound(Attr)
    SearchString = Attr(iAttr)
    Col_Attr(iAttr) = z_GetColumnIndex(SearchString, 1, Sh)
Next iAttr
For iCol_Attr = 0 To UBound(Col_Attr)
    Cells(2, Col_Attr(iCol_Attr)).EntireColumn.Select
    'change the date format
    Selection.NumberFormat = "dd-mmm-yyyy"
    'Selection.AutoFill Destination:=Range(Cells(2, Col_Attr(iCol_Attr)), Cells(RowSize,
Col_Attr(iCol_Attr))), Type:=xlFillDefault
    'perform a function (do not change them but add new ones)
    If Fkt = 1 Then
        For Row = 2 To RowSize
            Cells(Row, Col_Attr(iCol_Attr)) = Left(Cells(Row, Col_Attr(iCol_Attr)), 10)
            If (Row Mod 100) = 0 Then
                Debug.Print iCol_Attr & " " & Row
            End If
        Next Row
    End If
End Function

```

```
Next iCol_Attr
End Function
```

```
Function z_RenameCol(ColName_Old As String, ColName_New As String, Sh As String, Optional Row
As Long) As Integer
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Sheets(Sh).Activate
If Row = Empty Then
    Row = 1
End If
On Error GoTo ColName_Old_NotFound
Rows(Row).Find(What:=CStr(ColName_Old), LookAt:=xlWhole).Select
On Error GoTo 0
ActiveCell.Value = ColName_New
z_RenameCol = True
Exit Function
ColName_Old_NotFound:
    If z_ColExistent(ColName_New, Row, Sh) = True Then
        z_RenameCol = -1
    Else
        z_RenameCol = 0
    End If
End Function
```

```
Function z_ColExistent(ColName, Row, Sh) As Boolean
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
On Error GoTo ColName_Old_NotFound
Rows(Row).Find(What:=CStr(ColName), LookAt:=xlWhole).Select
z_ColExistent = True
On Error GoTo 0
Exit Function
ColName_Old_NotFound:
    z_ColExistent = False
End Function
```

```
Sub z_ChgFmt_CostCols(Sh As String, FromCol As Long, ToCol As Long)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 27.9.2011
Sheets(Sh).Select
RowSize = z_RowSize(1, Sh)
Range(Cells(2, FromCol), Cells(RowSize, ToCol)).Select
'Rows("1:1").Find(What:="EtcTrialsFullCosts2010", LookAt:=xlWhole).Select
'Range(ActiveCell.End(xlToRight).Offset(1, -2), ActiveCell.End(xlDown)).Select 'offset?
'Range(ActiveCell.End(xlToRight), ActiveCell.End(xlDown)).Select
Selection.NumberFormat = "#,##0"
Selection.Replace What:=".", Replacement:=".", LookAt:=xlPart, _
    SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
    ReplaceFormat:=False
Selection.ColumnWidth = 25
End Sub
```

Function z\_ListOfSortedAttributeEntries(ColName As String, Sh\_from As String, Row\_To As Long, Col\_To As Long, Sh\_to As String)

'Author: Franz Schuermann, Project Management Excellence

'Roland Benz, Project Management Excellence

'Date: 27.9.2011

'Input: "RD\_MasterDataSet"

'Output: "RD\_MasterDataSet", "Logfile\_CustomerCheck"

'Objective:

'Create a logfile with all unique entries in the attribute "ListOfTaskCustomers"

'Correct the wrong entries in the column "ListOfTaskCustomers" (user interaction)

'Create a logfile of tasks with wrong entries in the attribute "ListOfTaskCustomers"

'Give the list to the PIMs (user interaction)

Dim ColIndex As Long

Dim RowSize As Long

ColIndex = z\_GetColumnIndex(ColName, 1, Sh\_from)

RowSize = z\_RowSize(1, Sh\_from)

Call z\_CopyInsertRange(1, ColIndex, RowSize, ColIndex, Sh\_from, 1, 1, Sh\_to)

'set the filter and write out in column B all different entries in column A

'Range(Cells(1, 1), Cells(RowSize, 1)).AdvancedFilter Action:=xlFilterCopy, CopyToRange:=Range( \_  
"B1"), Unique:=True

Sheets(Sh\_to).Range(Cells(Row\_To, Col\_To), Cells(RowSize, Col\_To)).AdvancedFilter \_  
Action:=xlFilterCopy, CopyToRange:=Cells(Row\_To, Col\_To + 1), Unique:=True

'define the sort range

'ActiveWorkbook.Worksheets(Sh\_To).Sort.SortFields.Add Key:=Range("B2:B100"), \_  
SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:= \_  
xlSortNormal

Dim RowSize\_To As Long

RowSize\_To = z\_RowSize(Col\_To + 1, Sh\_to)

ActiveWorkbook.Worksheets("Log\_CustomerChk").Sort.SortFields.Clear

ActiveWorkbook.Worksheets(Sh\_to).Sort.SortFields.Add Key:=Range(Cells(Row\_To + 1, Col\_To + 1),  
Cells(RowSize\_To, Col\_To + 1)), \_  
SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:= \_  
xlSortNormal

'apply the sort within the sort range

With ActiveWorkbook.Worksheets(Sh\_to).Sort

.SetRange Range(Cells(Row\_To + 1, Col\_To + 1), Cells(RowSize\_To, Col\_To + 1))

.Header = xlYes

.MatchCase = False

.Orientation = xlTopToBottom

.SortMethod = xlPinYin

.Apply

End With

Cells(1, 2).Value = "z\_ListOfSortedAttributeEntries: " & Cells(1, 2).Value

Range("B1").EntireColumn.AutoFit

End Function

Function z\_StringSplit(Dltrlst As Variant, Rpllst As Variant, RowU\_From As Long, RowD\_From As Long, \_

Col\_From As Long, Sh\_from As String, RowU\_To As Long, Col\_To As Long, Sh\_to As String, \_  
Optional ByVal Wb As Workbook) As Variant

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

Sheets(Sh_from).Select
Call z_CopyPasteRange(RowU_From, Col_From, RowD_From, Col_From, Sh_from, RowU_To, Col_To,
Sh_to)
Sheets(Sh_to).Select
Dim Rng As Range
Set Rng = Range(Cells(RowU_To, Col_To), Cells(RowD_From, Col_To))
Rng.Select
'Die Zelle kopieren, und jeden String mit einem Zeichen trennen : leerezeichen,@ etc
'Columns("B:B").Select
For k = LBound(Dltrlst) To UBound(Dltrlst)
    If RplLst(k) = Empty Then
        l = RplLst(0)
    Else
        l = k
    End If
    Selection.Replace What:=Dltrlst(k), Replacement:=RplLst(l), LookAt:=xlPart, _
    SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
    ReplaceFormat:=False
Next k

Dim StringSplit_Dim1 As Variant
Dim NofCustomers As Integer
NofCustomers = z_NofCustomers(Rng, "@@@", Sh_to)
ReDim StringSplit_Dim2(RowU_From To RowD_From, Col_To + 1 To Col_To + 1 + (NofCustomers * 2))
As Variant
For i = RowU_To To RowD_From
    StringSplit_Dim1 = Split(Cells(i, Col_To), "@")
    p = 0
    For m = LBound(StringSplit_Dim1) To UBound(StringSplit_Dim1)
        If StringSplit_Dim1(m) <> Empty Then
            Cells(i, Col_To + p + 1) = StringSplit_Dim1(m)
            StringSplit_Dim2(i, Col_To + p + 1) = StringSplit_Dim1(m)
            p = p + 1
        End If
    Next m
Next i

z_StringSplit = StringSplit_Dim2
End Function

Function z_NofCustomers(Rng As Range, sLookupName As String, Sh As String) As Integer
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Dim Str As String, a
'Str = " andy,andy,tom,amy,andy"
'a = "andy"
'MsgBox UBound(Split(Str, a))
Dim sNames As String
Dim RngSize As Variant
RngSize = z_RangeSize(Rng, Sh)
ReDim scount(RngSize(1)) As Variant
For Row = 1 To RngSize(1) Step 1
    For Col = 1 To RngSize(2) Step 1
        sNames = Rng.Cells(Row, Col)
        scount(Row) = (Len(sNames) - Len(Replace(sNames, sLookupName, ""))) / Len(sLookupName)
    Next Col
Next Row

```

```

Next Col
Next Row
z_NoOfCustomers = Application.WorksheetFunction.Max(scount) + 1
End Function
Function z_RangeSize(Rng As Range, Sh As String) As Variant
Sheets(Sh).Select
Rng.Select
Dim RangeSize(1 To 4) As Long
RangeSize(1) = Rng.Rows.Count
RangeSize(2) = Rng.Columns.Count
RangeSize(3) = Rng.End(xlDown).Row
RangeSize(4) = Rng.End(xlToRight).Column
z_RangeSize = RangeSize
End Function
Function z_ListOfTaskCustomerError1(StringSplit_Dim2 As Variant, Optional Sh_log As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'filter the column "ListOfTaskCustomers" with values that make no sense (sum<>100%) and replace
them with 100%, replace unnecessary entries
'todo
'MsgBox ("Please clean the Customer list from illogical values and then resume the macro")
'Rows("1:1").Find(What:="ListOfTaskCustomers", LookAt:=xlWhole).Select
For Row = LBound(StringSplit_Dim2, 1) To UBound(StringSplit_Dim2, 1) Step 1
    For Col = LBound(StringSplit_Dim2, 2) To UBound(StringSplit_Dim2, 2) Step 1
        'IsNr = Application.WorksheetFunction.IsNumber(StringSplit_Dim2(row, col))
        IsNr = Application.WorksheetFunction.IsNumber(Cells(Row, Col))
        If IsNr <> False Then
            If Cells(Row, Col) < 0 Or Cells(Row, Col) > 100 Then
                Cells(Row, Col) = 100
                Cells(Row, Col).Font.Color = RGB(255, 0, 0)
                z_ListOfTaskCustomerError1 = False
            Else
                z_ListOfTaskCustomerError1 = True
            End If
        End If
    Next Col
Next Row
End Function
Function z_ListOfTaskCustomerError2(StringSplit_Dim2 As Variant, Sh_log As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'find double entries
Dim OneCustomer1 As String
Dim OneCustomer2 As String
For Row = LBound(StringSplit_Dim2, 1) To UBound(StringSplit_Dim2, 1) Step 1
    For Col = LBound(StringSplit_Dim2, 2) To UBound(StringSplit_Dim2, 2) Step 2
        'IsNrFlag = Application.WorksheetFunction.IsNumber(Cells(row, col))
        'OneCustomer1 = StringSplit_Dim2(row, col) & StringSplit_Dim2(row, col + 1)
        OneCustomer1 = Cells(Row, Col) & Cells(Row, Col + 1)
        For k = Col To UBound(StringSplit_Dim2, 2) - 2 Step 2
            If Cells(Row, k + 2) <> Empty Then
                OneCustomer2 = Cells(Row, k + 2) & Cells(Row, k + 3)
                If OneCustomer1 = OneCustomer2 Then
                    Stop
                    Cells(Row, k + 2).Delete Shift:=xlToLeft
                End If
            End If
        Next k
    Next Col
Next Row
End Function

```



```

        k = k - 1
        Cells(Row, k + 3).Delete Shift:=xlToLeft
        k = k - 1
    End If
End If
Next k
Next Col
Next Row
End Function
Function z_ListOfTaskCustomerError3(StringSplit_Dim2 As Variant, Sh_log As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'find strings that do not sum up to 100% and write these into a logfile for the PIMs
'todo
Dim Sh_dim As Variant
Sh_dim2 = z_LastWrittenRowAndCol(Sh_log, 500)
Sheets(Sh_log).Cells(1, Sh_dim2(1) + 1).Value = "z_ListOfTaskCustomerError3:Sum<>100or0"
Dim Sum As Double
For Row = LBound(StringSplit_Dim2, 1) + 1 To UBound(StringSplit_Dim2, 1) Step 1
    Sum = 0
    For Col = LBound(StringSplit_Dim2, 2) To UBound(StringSplit_Dim2, 2) Step 1
        'IsNr = Application.WorksheetFunction.IsNumber(StringSplit_Dim2(row, col))
        IsNr = Application.WorksheetFunction.IsNumber(Cells(Row, Col))
        If IsNr <> False Then
            Sum = Sum + Cells(Row, Col)
        End If
    Next Col
    If (Sum < 100.01 And Sum > 99.99) Or (Sum < 0.01 Or Sum > -0.01) Or Sum = Empty Then
        Sheets(Sh_log).Cells(Row, Sh_dim2(1) + 1).Value = Sum
    Else
        Sheets(Sh_log).Cells(Row, Sh_dim2(1) + 1).Value = "False"
    End If
Next Row
End Function

```

```

Function z_BuildSting(ByRef Rng_From As Range, Sh_from As String, ByRef Rng_To As Range, Sh_to
As String)
Sheets(Sh_from).Select
Dim Rng_Address As Variant
Rng_Address = z_RangeAddressToArray(Rng_From)
RowU = Rng_Address(1)
RowD = Rng_Address(3)
ColL = Rng_Address(2)
ColR = Rng_Address(4)
For Row = 1 To (RowD - RowU + 1)
    If Rng_From(Row, 3) = Empty Then
        Rng_To(Row, 1) = Rng_From(Row, 1) & "(" & Rng_From(Row, 2) & ".0%)"
    Else
        Rng_To(Row, 1) = Rng_From(Row, 1) & "(" & Rng_From(Row, 2) & ".0%)," & Rng_From(Row, 3) &
        "(" & Rng_From(Row, 4) & ".0%)"
    End If
Next Row
End Function

```

```
Function z_ReplaceWrongAttributEntries(Sh_from As String, ByRef Rng_What As Range, ByRef Rng_Repl As Range, _
```

```
    Sh_to As String, ByRef Rng_To As Range)
```

```
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

```
'Date: 27.9.2011
```

```
Dim Rng_Address As Variant
```

```
Rng_Address = z_RangeAddressAsArray(Rng_What)
```

```
RowU = Rng_Address(1)
```

```
RowD = Rng_Address(3)
```

```
ColL = Rng_Address(2)
```

```
ColR = Rng_Address(4)
```

```
For i = RowU To RowD
```

```
    Rng_What_i = Rng_What(i, 1)
```

```
    Rng_Repl_i = Rng_Repl(i, 1)
```

```
    If Rng_What_i <> Rng_Repl_i Then
```

```
        Sheets(Sh_to).Select
```

```
        Rng_To.Select
```

```
        Selection.Replace What:=Rng_What_i, Replacement:=Rng_Repl_i, LookAt:=xlPart, _
```

```
            SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
```

```
            ReplaceFormat:=False
```

```
    End If
```

```
Next i
```

```
Selection.ColumnWidth = 25
```

```
End Function
```

```
Sub t12()
```

```
Dim Rng_From As Range
```

```
Dim Rng_To As Range
```

```
'To set a range the sheet must be activated!
```

```
Sheets("Log_CustomerChk").Select
```

```
Set Rng_From = Sheets("Log_CustomerChk").Range(Cells(3, 4), Cells(47, 7))
```

```
Set Rng_To = Sheets("Log_CustomerChk").Range(Cells(3, 9), Cells(47, 9))
```

```
Sheets("Log_CustomerChk").Cells(1, 9) = "z_BuildSting"
```

```
Call z_BuildSting(Rng_From, "Log_CustomerChk", Rng_To, "Log_CustomerChk")
```

```
Dim Rng_What As Range
```

```
Dim Rng_Repl As Range
```

```
Dim Rng_To2 As Range
```

```
'To set a range the sheet must be activated!
```

```
Sheets("Log_CustomerChk").Select
```

```
Set Rng_What = Sheets("Log_CustomerChk").Range(Cells(3, 2), Cells(47, 2))
```

```
Set Rng_Repl = Sheets("Log_CustomerChk").Range(Cells(3, 9), Cells(47, 9))
```

```
'To set a range the sheet must be activated!
```

```
Sheets("RD_MasterDataSet4").Select
```

```
Set Rng_To2 = Sheets("RD_MasterDataSet4").Range(Cells(2, 45), Cells(74427, 45))
```

```
Call z_ReplaceWrongAttributEntries("Log_CustomerChk", Rng_What, Rng_Repl,
```

```
"RD_MasterDataSet4", Rng_To2)
```

```
End Sub
```

```
Function z_Split_Concatenated_ListOfTastCustomers(Sh As String, ColName As String)
```

```
'Macro generated by Franz.Schuermann@Syngenta.com and
```

```
'Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

'Date: 27.9.2011  
'Input: "RD\_MasterDataSet"  
'Output: "RD\_MasterDataSet"  
'Objective: Split concatenated string in Attribute "ListOfTaskCustomers" and store it in  
"CustomerName" and "Customer%"

'Ignore Task Customer lines for calculation with the following conditions:

- ' - 100%
- ' - 0%
- ' - Customer with a prefix XXX\_

'But at the same time remove the percentages and additional possible customers if:

- ' - Second customer has 0% or additional 100%
- ' - Customer with a prefix XXX\_

'Activate sheet and find column ColName

Sheets(Sh).Activate

Rows("1:1").Find(What:=ColName, LookAt:=xlWhole).Select

'change the format of the column "Customer%"

ActiveCell(1, 2).EntireColumn.NumberFormat = "#,##0"

'select range of column ColName

Range(ActiveCell(2, 1), ActiveCell.Offset(0, -44).End(xlDown).Offset(0, 44)).Select

'take each entry rCustomer of the column ColName

For Each rCustomer In Selection.Cells

    If Not rCustomer.Value = "" Then

        'remove all entries that start with XXX

        If Left(rCustomer, 4) = "XXX\_" Then

            rCustomer.Value = ""

        End If

    'if 100.0% is found write 100 into the left column same row

    ' split fkt with flag -1 cuts away everything left of " ("

    If Not rCustomer.Find(What:="100.0%", LookAt:=xlPart) Is Nothing Then

        rCustomer.Offset(0, 1).Value = "100"

        rCustomer.Value = Split(rCustomer, " (", -1) '!!!!

        'rCustomer.Value = InStr(1, rCustomer, Chr(40))

    End If

    'if no comma ", " is found

    ' if 0.0% is found write 0 into the left column same row

    ' split fkt with flag -1 cuts away everything left of " ("

    If rCustomer.Find(What:=",", LookAt:=xlPart) Is Nothing Then

        If Not rCustomer.Find(What:="(0.0%", LookAt:=xlPart) Is Nothing Then

            rCustomer.Offset(0, 1).Value = "0"

            rCustomer.Value = Split(rCustomer, " (", -1)

        End If

    End If

    'if a comma ", " is found

    ' if 0.0% is found write 0 into the left column same row

    ' split fkt with flag -1 cuts away everything left of " ("

```

If Not rCustomer.Find(What:=",", LookAt:=xlPart) Is Nothing Then
    If rCustomer.Find(What:="(0.0%)", LookAt:=xlPart) Is Nothing Then
        'insert a row
        rCustomer.Offset(1, 0).EntireRow.Insert
        'copy the values above
        rCustomer.EntireRow.Copy
        'paste the copied row values into the new row and!
        'split the string at the comma "" leave the left side in rCustomer
        'and paste the right side into the left column same row
        rCustomer.Offset(1, 0).EntireRow.PasteSpecial Paste:=xlPasteValues
        rCustomer.TextToColumns Destination:=rCustomer, DataType:=xlDelimited, _
            ConsecutiveDelimiter:=False, Other:=True, OtherChar:=", "
        'copy the value of the left column same row to the row below rcustomer
        rCustomer.Offset(1, 0).Value = rCustomer.Offset(0, 1).Value
        'left column same row leave the number and cut away the rest
        rCustomer.Offset(0, 1).Value = Mid(rCustomer.Value, InStr(1, rCustomer, "(") + 1, 3)
        'left column row below leave the number and cut away the rest
        rCustomer.Offset(1, 1).Value = Mid(rCustomer.Offset(1, 0).Value, InStr(1,
rCustomer.Offset(1, 0), "(") + 1, 3)
        'rcustomer cut away everything left of the " ("
        rCustomer.Value = Split(rCustomer, " (", -1)
        'row below rcustomer cut away everything left of the " ("
        rCustomer.Offset(1, 0).Value = Split(rCustomer.Offset(1, 0), " (", -1)
    End If
End If
End If
Next
End Function

```

Function z\_CostReduction\_ListOfTastCustomers(Sh As String, ColName\_Weight As String,  
ColName\_Start As String)  
'Macro generated by Franz.Schuermann@Syngenta.com (PMEC, Project Management Excellence)

```

'Activate sheet and find column indices
Sheets(Sh).Activate
Dim Index_Weight As Long
Dim Index_Start As Long
Dim Index_offset As Long
Index_Weight = z_GetColumnIndex(ColName_Weight, 1, Sh)
Index_Start = z_GetColumnIndex(ColName_Start, 1, Sh)
Index_offset = Index_Start - Index_Weight
'select column ColName and then select range of column ColName
Rows("1:1").Find(What:=ColName_Weight, LookAt:=xlWhole).Select
Range(ActiveCell(2, 1), ActiveCell.Offset(0, -45).End(xlDown).Offset(0, 45)).Select

```

```

For Each rPercentage In Selection.Cells
    'If the value in the row ColName is between between 0 and 100 (not 0 and not 100)
    If rPercentage.Value > 0 Then
        If rPercentage.Value < 100 Then
            'Select the range with the costs and calculate the weighted costs
            Range(rPercentage.Offset(0, Index_offset), rPercentage.Offset(0, Index_offset + 125)).Select
            For Each rFigure In Selection.Cells
                rFigure.Value = rPercentage.Value / 100 * rFigure.Value
            Next
        End If
    End If
Next

```

```

        Next
    End If
End If
Next

```

```

End Function

```

```

Function z_RemWholeProject_WithAttrEntryOnPiLev(Sh As String, Sh_log As String, RefColName As
String, _
    AttrColName As String, SearchAttrValue As String, Optional fkt_flag As Integer)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 12.10.2011
'Activate sheet and find column indices
Sheets(Sh).Activate
Dim Index_Ref As Long
Dim Index_Attr As Long
Index_Ref = z_GetColumnIndex(RefColName, 1, Sh)
Index_Attr = z_GetColumnIndex(AttrColName, 1, Sh)
'Look for the AttrValue and delete the entire project. Write out the row into Sh_log
Dim Row As Long: Row = 2
Dim ilog As Long: ilog = 2
Do Until Cells(Row, Index_Ref) = ""
    If fkt_flag = 0 Then
        AttrValue = Cells(Row, Index_Attr)
    End If
    'do not change the string function inside the UDF if necessary, but add new ones with new flags
    If fkt_flag = 1 Then
        AttrValue = Left(Cells(Row, Index_Attr), 2) 'change the function if necessary or add others with
new flag
    End If
    'Delete the whole project if the condition on Pi Level is fulfilled
    '    attribute value on levels below Pi are not checked!
    If AttrValue = SearchAttrValue Then
        Do Until Cells(Row, Index_Ref) <> Cells(Row + 1, Index_Ref)
            If 1 Then
                'Write out into the logfile
                Cells(Row, Index_Ref).EntireRow.Copy _
                    Destination:=Sheets(Sh_log).Cells(ilog, 1).EntireRow: ilog = ilog + 1
            End If
            'Delete the entire row
            Cells(Row, Index_Ref).EntireRow.Delete Shift:=xlUp
            'Row = Row - 1
        Loop
        If 1 Then
            'Write out into the logfile
            Cells(Row, Index_Ref).EntireRow.Copy _
                Destination:=Sheets(Sh_log).Cells(ilog, 1).EntireRow: ilog = ilog + 1
        End If
        'Delete the entire row
        Cells(Row, Index_Ref).EntireRow.Delete Shift:=xlUp
        'Row = Row - 1
    End If

```

```

Else
    Do Until Cells(Row, Index_Ref) <> Cells(Row + 1, Index_Ref)
        Row = Row + 1
    Loop
    Row = Row + 1
End If
Loop
Sheets(Sh_log).Cells(1, 1) = "z_RemRow_WithAttrEntry"

End Function

Sub test9()
'do not change the string function inside the UDF if necessary, but add new ones with new flags
Call z_RemWholeProject_WithAttrEntryOnPiLev("Sheet1", "Sheet2", _
    "PiIdentifier", "SyngentaPortfolioLevel1", "SEEDS")
Call z_RemRow_WithAttrEntry("Sheet1", "Sheet2", _
    "PiIdentifier", "ActivityIdentifier", "TK", 1)
End Sub

Function z_RemRow_WithAttrEntry(Sh As String, Sh_log As String, RefColName As String, _
    AttrColName As String, SearchAttrValue As String, Optional fkt_flag As Integer)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 12.10.2011
'Activate sheet and find column indices
Sheets(Sh).Activate
Dim Index_Ref As Long
Dim Index_Attr As Long
Index_Ref = z_GetColumnIndex(RefColName, 1, Sh)
Index_Attr = z_GetColumnIndex(AttrColName, 1, Sh)
'Look for the AttrValue and delete the entire row. Write out the row into Sh_log
Dim AttrValue As Variant
Dim Row As Long: Row = 2
Dim ilog As Long: ilog = 2
Do Until Cells(Row, Index_Ref) = ""
    If fkt_flag = 0 Then
        AttrValue = Cells(Row, Index_Attr)
    End If
    'do not change the string function inside the UDF if necessary, but add new ones with new flags
    If fkt_flag = 1 Then
        AttrValue = Left(Cells(Row, Index_Attr), 2)
    End If
    If AttrValue = SearchAttrValue Then
        If 1 Then
            'Write out into the logfile
            Cells(Row, Index_Ref).EntireRow.Copy _
                Destination:=Sheets(Sh_log).Cells(ilog, 1).EntireRow: ilog = ilog + 1
        End If
        'Delete the entire row
        Cells(Row, Index_Ref).EntireRow.Delete Shift:=xlUp
        Row = Row - 1
    End If
    Row = Row + 1
Loop

```

```
Sheets(Sh_log).Cells(1, 1) = "z_RemRow_WithAttrEntry"  
End Function
```

```
Function z_CpyAttrEntryOnPiLev_ToLevsBelow(Sh As String, Pild As String, ActId As String, _  
    AttrColName As String)
```

```
End Function
```

```
Sub test()  
Dim ArrayCost1(5000, 600) As Double  
Dim ArrayLevel1(5000, 600) As String  
Dim ArrayRow1(5000, 600) As Long  
Call FillArrayCosts_PiNr_ActNr("RD_MasterDataSet7", 134, ArrayCost1)  
Call FillArrayPiLevel_PiNr_ActNr("RD_MasterDataSet7", 2, ArrayLevel1)  
Call FillArrayRow_PiNr_ActNr("RD_MasterDataSet7", ArrayRow1)  
Stop  
End Sub
```

```
Function z_FillArrayCosts_PiNr_ActNr(Sh As String, Col As Long, ByRef ArrayCost() As Double)  
Dim PiNr As Long  
Dim ActNr As Integer  
PiNr = 0 'one index nr for each projects  
ActNr = 0 'one index for each position of a project PiNr  
Sheets(Sh).Activate  
RowSize = IIf(IsEmpty(Range("A1048576")), Range("A1048576").End(xlUp).Row, 1048576)  
For Row = 2 To RowSize  
    'Start of a new project  
    ArrayCost(PiNr, ActNr) = Cells(Row, Col)  
    'if still the same project  
    If Cells(Row, 1) = Cells(Row + 1, 1) Then  
        ActNr = ActNr + 1  
        ArrayCost(PiNr, ActNr) = Cells(Row, Col)  
    Else  
        ActNr = 0  
        PiNr = PiNr + 1  
    End If  
Next Row  
End Function  
Function z_FillArrayPiLevel_PiNr_ActNr(Sh As String, Col_ActId As Long, ByRef ArrayLevel() As String)  
Dim PiNr As Long  
Dim ActNr As Integer  
PiNr = 0 'one index nr for each projects  
ActNr = 0 'one index for each position of a project PiNr  
Sheets(Sh).Activate  
RowSize = IIf(IsEmpty(Range("A1048576")), Range("A1048576").End(xlUp).Row, 1048576)  
For Row = 2 To RowSize  
    'Start of a new project  
    ArrayLevel(PiNr, ActNr) = Left(Cells(Row, Col_ActId), 2)  
    'if still the same project  
    If Cells(Row, 1) = Cells(Row + 1, 1) Then
```

```

        ActNr = ActNr + 1
        ArrayLevel(PiNr, ActNr) = Left(Cells(Row, Col_ActId), 2)
    Else
        ActNr = 0
        PiNr = PiNr + 1
    End If
Next Row
End Function
Function z_FillArrayRow_PiNr_ActNr(Sh As String, ByRef ArrayRow() As Long)
Dim PiNr As Long
Dim ActNr As Integer
PiNr = 0 'one index nr for each projects
ActNr = 0 'one index for each position of a project PiNr
Sheets(Sh).Activate
RowSize = IIf(IsEmpty(Range("A1048576")), Range("A1048576").End(xlUp).Row, 1048576)
For Row = 2 To RowSize
    'Start of a new project
    ArrayRow(PiNr, ActNr) = Row
    'if still the same project
    If Cells(Row, 1) = Cells(Row + 1, 1) Then
        ActNr = ActNr + 1
        ArrayRow(PiNr, ActNr) = Row
    Else
        ActNr = 0
        PiNr = PiNr + 1
    End If
Next Row
End Function

Function z_FillArrayActNr_PiNr(Sh As String, ByRef ArrayRow() As Long, ByRef ArrayActNr() As Long)
'Find the ActNr for each PiNr
PiNr = 0
ActNr = 0
Do Until ArrayRow(PiNr, 0) = 0
    Do Until ArrayRow(PiNr, ActNr) = 0
        ArrayActNr(PiNr) = ActNr 'Used to speed up the program
        ActNr = ActNr + 1
    Loop
    ActNr = 0
    PiNr = PiNr + 1
Loop
End Function

Function z_FindArraySizes(Sh As String, ByRef ArrayRow() As Long) As Variant
'Find the last PiNr and the highest ActNr
Dim out(0 To 1) As Long
PiNr = 0
ActNr = 0
ActNrMax = 0
Do Until ArrayRow(PiNr, 0) = 0
    Do Until ArrayRow(PiNr, ActNr) = 0
        If ActNr > ActNrMax Then
            ActNrMax = ActNr 'used to check whether the dimensions of the arrays are big enough

```



```

        'Poskmax = PiNr
    End If
    ActNr = ActNr + 1
    Loop
    PiNr = PiNr + 1
    Loop
    PiNrMax = PiNr 'used to speed up the program and used to check whether the dimensions of the
arrays are big enough
    out(0) = PiNrMax
    out(1) = ActNrMax
    z_FindArraySizes = out
End Function

```

```

Function z_RedimArrays(Sh As String, ByRef ArrayRow() As Long, ByRef ArrayActNr() As Long, ByRef
ArrayCost() As Double, _

```

```

    ByRef ArrayCostF() As Double, ByRef ArrayLevel() As String)

```

```

'if arrays are written to their limit try to redim

```

```

    Sheets(Sh).Activate

```

```

    Dim SomeMorePi As Integer

```

```

    Dim SomeMoreAct

```

```

    SomeMorePi = 0

```

```

    SomeMoreAct = 0

```

```

    ArraySizes = z_FindArraySizes(Sh, ArrayRow)

```

```

    Do Until UBound(ArrayRow, 1) > (ArraySizes(0) + 10)

```

```

        'more PIs

```

```

        ReDim ArrayRow(0 To 5000 + SomeMorePi, 0 To 600) As Long

```

```

        ReDim ArrayActNr(0 To 5000 + SomeMorePi) As Long

```

```

        ReDim ArrayCost(0 To 5000 + SomeMorePi, 0 To 600) As Double

```

```

        ReDim ArrayCostF(0 To 5000 + SomeMorePi, 0 To 600) As Double

```

```

        ReDim ArrayLevel(0 To 5000 + SomeMorePi, 0 To 600) As String

```

```

        Call z_FillArrayRow_PiNr_ActNr("RD_MasterDataSet7", ArrayRow)

```

```

        ArraySizes = z_FindArraySizes(Sh, ArrayRow)

```

```

        SomeMorePi = SomeMorePi + 20

```

```

    Loop

```

```

    SomeMoreAct = 0

```

```

    Do Until UBound(ArrayRow, 2) > (ArraySizes(1) + 10)

```

```

        'more Activities

```

```

        ReDim ArrayRow(0 To 5000 + SomeMorePi, 0 To 600 + SomeMoreAct) As Long

```

```

        ReDim ArrayActNr(0 To 5000 + SomeMorePi) As Long

```

```

        ReDim ArrayCost(0 To 5000 + SomeMorePi, 0 To 600 + SomeMoreAct) As Double

```

```

        ReDim ArrayCostF(0 To 5000 + SomeMorePi, 0 To 600 + SomeMoreAct) As Double

```

```

        ReDim ArrayLevel(0 To 5000 + SomeMorePi, 0 To 600 + SomeMoreAct) As String

```

```

        Call z_FillArrayRow_PiNr_ActNr("RD_MasterDataSet7", ArrayRow)

```

```

        ArraySizes = z_FindArraySizes(Sh, ArrayRow)

```

```

        SomeMoreAct = SomeMoreAct + 20

```

```

    Loop

```

```

End Function

```

```

Sub test13()

```

```

    Call EACRecalc("RD_MasterDataSet7", "Log_EACRecalc")

```

```

End Sub

```

```

Function z_EACRecalc(Sh As String, Sh_log As String)

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)  
'Date: 11.9.2011

'Program:

'logfile add column names

Sheets(Sh\_log).Activate

Dim ilog As Long: ilog = 1

\*\*\*0. Set the flag for Testmode. 1=on, 0=off\*\*\*

Dim test As Integer

test = 0 'flag for test purposes: 1=on, 0=off

\*\*\*1. Declaration and Initialisation\*\*\*

'Static arrays: Dynamic arrays and initialisation with ReDim causes memory problems at my laptop

ReDim ArrayRow(0 To 5000, 0 To 600) As Long

ReDim ArrayActNr(0 To 5000) As Long

ReDim ArrayCost(0 To 5000, 0 To 600) As Double

ReDim ArrayCostF(0 To 5000, 0 To 600) As Double 'data type double removes the 1.00d-2 problem  
from SmartChoice where the value is overwritten (all in test=1 mode)

ReDim ArrayLevel(0 To 5000, 0 To 600) As String 'variant

Dim ArraySizes As Variant

'Screen updating no

Application.ScreenUpdating = False

'Column ActivityIdentifier

'Activate sheet and find column indices

Sheets(Sh).Activate

Dim Col\_ActId As Long

Col\_ActId = z\_GetColumnIndex("ActivityIdentifier", 1, Sh) 'column used to extract the project level  
PI,MS,TK

'To measure the duration of the code: separate declaration of each variable. Otherwise it will not  
work out

Dim Start As Date: Dim StartDel As Date: Dim StartRecalcFull As Date: Dim StartRecalc20XX As Date:

Dim StartRemovingErrors As Date

Dim Duration As Date: Dim DurationDel As Date: Dim DurationRecalcFull As Date: Dim

DurationRecalc20XX As Date: Dim DurationRemovingErrors As Date

Start = Now() 'to measure the duration of the code

\*\*\*3. Recalculate the MS and PI cost residuals\*\*\*

If 1 Then 'flag: 1=on, 0=off

    'Activate sheet and find column indices

    Sheets(Sh).Activate

    Dim Col\_TotalFullCosts As Long

    Col\_TotalFullCosts = z\_GetColumnIndex("EacFullCosts", 1, Sh)

    Dim Col As Long

    Dim ColOut As Long

    For Col = Col\_TotalFullCosts To (Col\_TotalFullCosts - 5) Step -1 'packs of six columns

        If test Then

            ColOut = Col + 200 'write at the end of the spreadsheet

        Else

```

    ColOut = Col 'overwrite the columns
End If
'fill an ArrayRow(PiNr, ActNr) with the index Row,
'and an array ArrayLevel(PiNr, ActNr) with the project level info (PI,WS,TK)
PiNr = 0 'one index nr for each projects
ActNr = 0 'one index for each position of a project PiNr
RowSize = If(IsEmpty(Range("A1048576")), Range("A1048576").End(xlUp).Row, 1048576)

'Fill the arrays
If Col = Col_TotalFullCosts Then 'EACFullCosts (Separate calculation of EACFullCosts to speed up
the program)
    StartRecalcFull = Now() 'to measure the duration of the code

    'Initialize the EACFullCost Array with empty values (just in case there are any others)
    'Watch out when the dimensions are no longer big enough:
    ' more than 5000 PI or more than 600 WS and TK per one PI
    Call z_FillArrayRow_PiNr_ActNr(Sh, ArrayRow)

    'Check whether a redim is necessary and if so enhance the array sizes
    Call z_RedimArrays(Sh, ArrayRow, ArrayActNr, ArrayCost, ArrayCostF, ArrayLevel)

    'fill the array
    Call z_FillArrayActNr_PiNr(Sh, ArrayRow, ArrayActNr)
    Call z_FillArrayCosts_PiNr_ActNr(Sh, Col_TotalFullCosts, ArrayCostF)
    Call z_FillArrayPiLevel_PiNr_ActNr(Sh, Col_ActId, ArrayLevel)

    'determine the RiNrMax
    Dim PiNrMax As Long
    Dim ActNrMax As Long
    Dim MaxIter As Variant
    MaxIter = z_FindArraySizes(Sh, ArrayRow)
    PiNrMax = MaxIter(0)
    ActNrMax = MaxIter(1)

    'Calculate the residuals of the PI and WS levels
    For PiNr = 0 To PiNrMax
        If ArrayCost(PiNr, 0) <> 0 Or ArrayCostF(PiNr, 0) <> 0 Then 'Costs on PI level zero
            WSTK = 0 'sums all TKs of one WS
            PITK = 0 'sums all TKs of one PI
            PIWSres = 0 'sums all residual WSs of one PI
            WSres = 0 'residual of one WS
            Pires = 0 'residual of the PI
            For ActNr = ArrayActNr(PiNr) To 0 Step -1
                If ArrayCostF(PiNr, ActNr) <> 0 Then 'Cost on any level
                    If ArrayLevel(PiNr, ActNr) = "" Then
                        'do nothing and count down
                    Else
                        If ArrayLevel(PiNr, ActNr) = "TK" Then
                            If test Then
                                'Write nothing if test=0
                                Cells(ArrayRow(PiNr, ActNr), ColOut) = ArrayCostF(PiNr, ActNr) 'for test
purposes!!!
                            End If
                        End If
                    End If
                End If
            Next ActNr
        End If
    Next PiNr

```

```

WSTK = WSTK + ArrayCostF(PiNr, ActNr) 'WSTK = WSTK + Cells(ArrayRow(PiNr,
ActNr), Col)
PITK = PITK + ArrayCostF(PiNr, ActNr) 'PITK = PITK + Cells(ArrayRow(PiNr, ActNr),
Col)
Elseif ArrayLevel(PiNr, ActNr) = "WS" Then
    If WSTK = 0 Then
        If test Then
            'Write nothing if test=0
            Cells(ArrayRow(PiNr, ActNr), ColOut) = ArrayCostF(PiNr, ActNr) 'for test
purposes!!!
        End If
        PIWSres = PIWSres + ArrayCostF(PiNr, ActNr) 'PIWSres = PIWSres +
Cells(ArrayRow(PiNr, ActNr), Col)
        WSTK = 0
    Else
        'Overwrite
        WSres = ArrayCostF(PiNr, ActNr) - WSTK
        If WSres > -0.001 And WSres < 0.001 Then 'to remove thousands of rounding
errors from SmartChoice
            Cells(ArrayRow(PiNr, ActNr), ColOut) = 0
        Else
            Cells(ArrayRow(PiNr, ActNr), ColOut) = WSres
        End If
        PIWSres = PIWSres + ArrayCostF(PiNr, ActNr) - WSTK 'PIWSres = PIWSres +
Cells(ArrayRow(PiNr, ActNr), Col) - WSTK
        WSTK = 0
    End If
Elseif ArrayLevel(PiNr, ActNr) = "PI" Then
    'Overwrite
    Pires = ArrayCostF(PiNr, ActNr) - PIWSres - PITK
    If Pires > -0.001 And Pires < 0.001 Then 'to remove thousands of rounding errors
from SmartChoice
        Cells(ArrayRow(PiNr, ActNr), ColOut) = 0
    Else
        Cells(ArrayRow(PiNr, ActNr), ColOut) = Pires
    End If
Else
    Stop 'Wrong level, other than "PI", "WS", "TK"
End If
End If
End If
Next ActNr
End If
Next PiNr
DurationRecalcFull = Now() - StartRecalcFull 'to measure the duration of the code

Else 'EACCOSTS20XX
If Col = (Col_TotalFullCosts - 1) Then
    StartRecalc20XX = Now() 'to measure the duration of the code
End If

'fill the cost array
Call z_FillArrayCosts_PiNr_ActNr(Sh, Col, ArrayCost)

```

```

'Calculate the residualsof the PI and WS levels
For PiNr = 0 To PiNrMax
  If ArrayCost(PiNr, 0) <> 0 Or ArrayCostF(PiNr, 0) <> 0 Then 'Costs on PI level is zero
    WSTK = 0 'sums all TKs of one WS
    PITK = 0 'sums all TKs of one PI
    PIWSres = 0 'sums all residual WSs of one PI
    WSres = 0 'residual of one WS
    PIres = 0 'residual of the PI
    For ActNr = ArrayActNr(PiNr) To 0 Step -1
      If ArrayCost(PiNr, ActNr) <> 0 Or ArrayCostF(PiNr, ActNr) <> 0 Then 'Cost on any level is
zero
        If ArrayLevel(PiNr, ActNr) = "" Then
          'do nothing and count down
        Else
          If ArrayLevel(PiNr, ActNr) = "TK" Then
            If test Then
              'Write nothing if test=0
              Cells(ArrayRow(PiNr, ActNr), ColOut) = ArrayCost(PiNr, ActNr) 'for test
purposes!!!
            End If
            WSTK = WSTK + ArrayCost(PiNr, ActNr) 'WSTK = WSTK + Cells(ArrayRow(PiNr,
ActNr), Col)
            PITK = PITK + ArrayCost(PiNr, ActNr) 'PITK = PITK + Cells(ArrayRow(PiNr, ActNr),
Col)
          ElseIf ArrayLevel(PiNr, ActNr) = "WS" Then
            If WSTK = 0 Then
              If test Then
                'Write nothing if test=0
                Cells(ArrayRow(PiNr, ActNr), ColOut) = ArrayCost(PiNr, ActNr) 'for test
purposes!!!
              End If
              PIWSres = PIWSres + ArrayCost(PiNr, ActNr) 'PIWSres = PIWSres +
Cells(ArrayRow(PiNr, ActNr), Col)
              WSTK = 0
            Else
              'Overwrite
              WSres = ArrayCost(PiNr, ActNr) - WSTK
              If WSres > -0.001 And WSres < 0.001 Then 'to remove thousands of rounding
errors from SmartChoice
                Cells(ArrayRow(PiNr, ActNr), ColOut) = 0
              Else
                Cells(ArrayRow(PiNr, ActNr), ColOut) = WSres
              End If
              PIWSres = PIWSres + ArrayCost(PiNr, ActNr) - WSTK 'PIWSres = PIWSres +
Cells(ArrayRow(PiNr, ActNr), Col) - WSTK
              WSTK = 0
            End If
          ElseIf ArrayLevel(PiNr, ActNr) = "PI" Then
            'Overwrite
            PIres = ArrayCost(PiNr, ActNr) - PIWSres - PITK
            If PIres > -0.001 And PIres < 0.001 Then 'to remove thousands of rounding errors
from SmartChoice

```

```

        Cells(ArrayRow(PiNr, ActNr), ColOut) = 0
    Else
        Cells(ArrayRow(PiNr, ActNr), ColOut) = Plres
    End If
Else
    Stop 'Wrong level, other than "PI", "WS", "TK"
End If
End If
End If
Next ActNr
End If
Next PiNr
End If
Next Col
DurationRecalc20XX = Now() - StartRecalc20XX 'to measure the duration of the code
End If 'flag: 1=on, 0=off

***4. Find errors and known problems. Correct them where possible. Write them into the logfile.
Write the durations of each part into the logfile***
If 1 Then 'flag: 1=on, 0=off
    Sheets(Sh).Activate
    StartRemovingErrors = Now() 'to measure the duration of the code
    ilogStart = ilog + 1: ilog = ilog + 2

    For Col = Col_TotalFullCosts To (Col_TotalFullCosts - 5) Step -1
        If test Then
            ColOut = Col + 200
        Else
            ColOut = Col
        End If
        Call z_ReplNearZeroWithZero(Sh, Sh_log, ColOut, ilog) 'this task is now already performed above
        Call z_ReplIsTextWithZero(Sh, Sh_log, ColOut, ilog)
        Call FindNeativeValues(Sh, Sh_log, ColOut, ilog)
    Next Col

    DurationRemovingErrors = Now() - StartRemovingErrors 'to measure the duration of the code
    Sheets(Sh_log).Cells(ilogStart, 1) = "Errors and Corrections" & CStr(DurationRemovingErrors)
End If 'flag: 1=on, 0=off

'Screen updating Yes
Application.ScreenUpdating = True

'Write the durations into the logfile
Duration = Now() - Start
Sheets(Sh_log).Cells(ilog + 2, 1) = "Duration" & CStr(Duration): ilog = ilog + 1
Sheets(Sh_log).Cells(ilog, 1) = "DurationDel" & CStr(DurationDel): ilog = ilog + 1
Sheets(Sh_log).Cells(ilog, 1) = "DurationRecalcFull" & CStr(DurationRecalcFull): ilog = ilog + 1
Sheets(Sh_log).Cells(ilog, 1) = "DurationRecalc20XX" & CStr(DurationRecalc20XX): ilog = ilog + 1
Sheets(Sh_log).Cells(ilog, 1) = "DurationRemovingErrors" & CStr(DurationRemovingErrors): ilog = ilog
+ 1
'MsgBox ("Duration=" & Duration & Chr(10) & "ActNrMax(<600)=" & ActNrMax & Chr(10) &
"PiNrMax(<5000)=" & PiNrMax)
'Stop 'for test purposes (to scrutinize the final values of the variables)

```

End Function

Function z\_ReplNearZeroWithZero(Sh As String, Sh\_log As String, ColOut As Long, ilog As Long)

Row = 2

Do Until Cells(Row, 1) = ""

    If Cells(Row, ColOut) <> "" And Cells(Row, ColOut) <> 0 Then

        'replace positive values near zero with zero

        If Cells(Row, ColOut) > -0.001 And Cells(Row, ColOut) < 0.001 Then

            'Write out into the logfile

            Cells(Row, ColOut).EntireRow.Copy Destination:=Sheets(Sh\_log).Cells(ilog, 1).EntireRow

            Sheets(Sh\_log).Cells(ilog, 160) = "1: costs near zero": ilog = ilog + 1

        'Correction

        Cells(Row, ColOut) = 0

    End If

End If

Row = Row + 1

Loop

End Function

Function z\_ReplIsTextWithZero(Sh As String, Sh\_log As String, ColOut As Long, ilog As Long)

Row = 2

Do Until Cells(Row, 1) = ""

    If Cells(Row, ColOut) <> "" And Cells(Row, ColOut) <> 0 Then

        'replace 1.00d-2 with zero

        If WorksheetFunction.IsText(Cells(Row, ColOut)) Then

            'Write out into the logfile

            Cells(Row, ColOut).EntireRow.Copy Destination:=Sheets(Sh\_log).Cells(ilog, 1).EntireRow

            Sheets(Sh\_log).Cells(ilog, 160) = "2: costs IsText": ilog = ilog + 1

        'Correction

        Cells(Row, ColOut) = 0

    End If

End If

Row = Row + 1

Loop

End Function

Function FindNeativeValues(Sh As String, Sh\_log As String, ColOut As Long, ilog As Long)

Row = 2

Do Until Cells(Row, 1) = ""

    If Cells(Row, ColOut) <> "" And Cells(Row, ColOut) <> 0 Then

        'look for negative values

        If Cells(Row, ColOut) < -0.001 Then

            'Write out into the logfile

            Cells(Row, ColOut).EntireRow.Copy Destination:=Sheets(Sh\_log).Cells(ilog, 1).EntireRow

            Sheets(Sh\_log).Cells(ilog, 160) = "3: costs negative": ilog = ilog + 1

        'Correction: go into the logfile and

    End If

End If

Row = Row + 1

Loop

End Function

Sub z\_ComparisonOnPILevel(Sh\_BeforeRecalc As String, Sh\_AfterRecalc As String, Sh\_log As String)

'Author: Roland Benz, Project Management & Excellence

'Date: 13.9.2011

'Input:

'Output: Sh\_log

'Objective:

'Check the results of the Macro "z\_EACRecalc()"

'by adding the costs from the Sheet Sh\_BeforeRecalc

'by adding the costs from the Sheet Sh\_AfterRecalc

'Main tasks:

'\*\*\*1. Write out all PIs, and build a sum over all PI cost

'\*\*\*2. Sum up the from the recalculated costs for each project, write the PIs out and build a sum over all PI costs

Dim Start1 As Date: Dim Duration1 As Date

Dim Start2 As Date: Dim Duration2 As Date

'Screen updating no

Application.ScreenUpdating = False

'Write/label the columns with names

If 1 Then

Sheets(Sh\_log).Activate

Sheets(Sh\_log).Cells(1, 1) = "PIIdentifier"

Sheets(Sh\_log).Cells(1, 2) = "ActivityIdentifier"

Sheets(Sh\_log).Cells(1, 3) = "EacFullCosts2010"

Sheets(Sh\_log).Cells(1, 4) = "EacFullCosts2011"

Sheets(Sh\_log).Cells(1, 5) = "EacFullCosts2012"

Sheets(Sh\_log).Cells(1, 6) = "EacFullCosts2013"

Sheets(Sh\_log).Cells(1, 7) = "EacFullCosts2014"

Sheets(Sh\_log).Cells(1, 8) = "EacFullCosts"

Sheets(Sh\_log).Cells(1, 10) = "PIIdentifier\_2"

Sheets(Sh\_log).Cells(1, 11) = "ActivityIdentifier\_2"

Sheets(Sh\_log).Cells(1, 12) = "EacFullCosts2010\_2"

Sheets(Sh\_log).Cells(1, 13) = "EacFullCosts2011\_2"

Sheets(Sh\_log).Cells(1, 14) = "EacFullCosts2012\_2"

Sheets(Sh\_log).Cells(1, 15) = "EacFullCosts2013\_2"

Sheets(Sh\_log).Cells(1, 16) = "EacFullCosts2014\_2"

Sheets(Sh\_log).Cells(1, 17) = "EacFullCosts\_2"

Rows(1).Select

Selection.EntireColumn.AutoFit

End If

'\*\*\*1. Write out all PIs, and build a sum over all PI costs

If 1 Then

Start1 = Now()

Dim Col\_ActId As Long

Col\_ActId = z\_GetColumnIndex("ActivityIdentifier", 1, Sh\_BeforeRecalc) 'ActivityIdentifier column

'Activate sheet and find column indices

Sheets(Sh\_BeforeRecalc).Activate

Dim Col\_TotalFullCosts As Long

Col\_TotalFullCosts = z\_GetColumnIndex("EacFullCosts", 1, Sh\_BeforeRecalc)

Col\_FirstFullCost = Col\_TotalFullCosts - 5 'first EAC column



```

Row = 2 'run down the rows
'logfile iterator
ilog = 2 'used to increment after a new line was written into the logfile
Do Until Cells(Row, Col_ActId) = ""
    PILEv = Left(Cells(Row, Col_ActId), 2)
    If PILEv = "PI" Then
        If 1 Then
            'Write out into the logfile
            Sheets(Sh_log).Cells(ilog, 1) = Cells(Row, 1)
            Sheets(Sh_log).Cells(ilog, 2) = Cells(Row, Col_ActId)
            Range(Cells(Row, Col_FirstFullCost), Cells(Row, Col_FirstFullCost + 5)).Copy
            Application.Sheets(Sh_log).Cells(ilog, 3).PasteSpecial xlPasteValues
            ilog = ilog + 1
        End If
    End If
    Row = Row + 1
Loop
Duration1 = Now() - Start1 'to measure the duration of the code

'make a sum of all PIs
Sheets(Sh_log).Activate
RowSize = If(IsEmpty(Range("A1048576")), Range("A1048576").End(xlUp).Row, 1048576)
For Col = 3 To 8
    Cells(RowSize + 2, Col) = Application.Sum(Range(Cells(2, Col), Cells(RowSize, Col)))
Next Col
End If

```

```

***2.Sum up the from the recalculated costs for each project, write the PIs out and build a sum over
all PI costs
If 1 Then
    Sheets(Sh_AfterRecalc).Activate
    Start2 = Now()
    'Sum up from the recalculated costs for each project
    Col_ActId = z_GetColumnIndex("ActivityIdentifier", 1, Sh_AfterRecalc) 'ActivityIdentifier column
    'Activate sheet and find column indices
    Col_TotalFullCosts = z_GetColumnIndex("EacFullCosts", 1, Sh_AfterRecalc)
    Col_FirstFullCost = Col_TotalFullCosts - 5 'first EAC column
    Row = 2 'run down the rows
    'logfile iterator
    ilog = 2 'used to increment after a new line was written into the logfile
    Do Until Cells(Row, Col_ActId) = ""
        ifirst = Row
        EAC10 = Cells(Row, Col_FirstFullCost)
        EAC11 = Cells(Row, Col_FirstFullCost + 1)
        EAC12 = Cells(Row, Col_FirstFullCost + 2)
        EAC13 = Cells(Row, Col_FirstFullCost + 3)
        EAC14 = Cells(Row, Col_FirstFullCost + 4)
        EACFull = Cells(Row, Col_FirstFullCost + 5)
        Do Until Cells(Row, 1) <> Cells(Row + 1, 1)
            'Sum up
            EAC10 = EAC10 + Cells(Row + 1, Col_FirstFullCost)
            EAC11 = EAC11 + Cells(Row + 1, Col_FirstFullCost + 1)

```

```

EAC12 = EAC12 + Cells(Row + 1, Col_FirstFullCost + 2)
EAC13 = EAC13 + Cells(Row + 1, Col_FirstFullCost + 3)
EAC14 = EAC14 + Cells(Row + 1, Col_FirstFullCost + 4)
EACFull = EACFull + Cells(Row + 1, Col_FirstFullCost + 5)
Row = Row + 1
Loop
'Write out
Sheets(Sh_log).Cells(ilog, 10) = Cells(ifirst, 1)
Sheets(Sh_log).Cells(ilog, 11) = Cells(ifirst, Col_ActId)
Sheets(Sh_log).Cells(ilog, 12) = EAC10
Sheets(Sh_log).Cells(ilog, 13) = EAC11
Sheets(Sh_log).Cells(ilog, 14) = EAC12
Sheets(Sh_log).Cells(ilog, 15) = EAC13
Sheets(Sh_log).Cells(ilog, 16) = EAC14
Sheets(Sh_log).Cells(ilog, 17) = EACFull
ilog = ilog + 1
Row = Row + 1
Loop
Duration2 = Now() - Start2 'to measure the duration of the code

'make a sum of all PIs
Sheets(Sh_log).Activate
RowSize = If(IsEmpty(Range("J1048576")), Range("J1048576").End(xlUp).Row, 1048576)
For q = 12 To 17
    Cells(RowSize + 2, q) = Application.Sum(Range(Cells(2, q), Cells(RowSize, q)))
Next q
End If

'adds some additional info and some format changes

'Formats
Cells.Select
Selection.NumberFormat = "#,##0"
Columns(9).ColumnWidth = 3

'Fill in formulas to calculate differences
Range("S2").Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "=RC[-16]-RC[-7]"
Selection.AutoFill Destination:=Range("S2:X2"), Type:=xlFillDefault
Range("S2:X2").Select
Selection.Copy
RowSize = z_RowSize(1, Sh_log)
Range("S2:X" & RowSize).Select
ActiveSheet.Paste

'Copy the Col Names
Range("L1:Q1").Select
Selection.Copy
Range("S1").Select
ActiveSheet.Paste

'Select the filter

```

```
Rows("1:1").Select
Range("M1").Activate
Application.CutCopyMode = False
Selection.AutoFilter
```

```
'Freeze panel
Range("C2").Select
ActiveWindow.FreezePanels = True
```

```
'Screen updating Yes
Application.ScreenUpdating = True
```

```
'Write the durations into the logfile
Sheets(Sh_log).Cells(ilog + 2, 1) = "Duration" & CStr(Duration1): ilog = ilog + 1
Sheets(Sh_log).Cells(ilog, 1) = "DurationDel" & CStr(Duration2): ilog = ilog + 1
End Sub
```

```
'Mit online report auf Projektebene vergleichen (Von Franz S. erzeugt)
Sub z_CostComparison_GenerateSmCExtract()
```

```
End Sub
```

```
Sub z_CostComparisonWithSmCExtract()
```

```
End Sub
```

```
Sub z_CostsTest(Sh_from As String, Sh_to As String, Row_From As Long, Row_To As Long)
```

```
'copy the costs column
Sheets(Sh_from).Select
Range("BQ" & CStr(Row_From) & ":" & "GL" & CStr(Row_From)).Select
Selection.Copy
'paste them
Sheets(Sh_to).Select
Cells(Row_To, 1).Select
Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _
    False, Transpose:=True
'write them at the right position: make six packs
RowSize = z_RowSize(1, Sh_to)
Col = 5
For Row = Row_To To RowSize Step 6
    Range(Cells(Row, 1), Cells(Row + 5, 1)).Select
    Selection.Cut
    Cells(Row_To, Col).Select
    ActiveSheet.Paste
    Col = Col + 1
Next Row
'Add the PI Info
'copy the costs column
Sheets(Sh_from).Select
Range("A" & CStr(Row_From) & ":" & "B" & CStr(Row_From)).Select
Selection.Copy Destination:=Sheets(Sh_to).Range("B" & CStr(Row_To) & ":" & "C" & CStr(Row_To))
```

```

    Sheets(Sh_to).Select
    Cells(1, 1).Select
End Sub

```

```

Function z_CostTestFormating(Sh_from As String, Sh_to As String, Row_From As Long, NrOfRows As Long)

```

```

    'Clear All
    Sheets(Sh_to).Select
    Cells.Select
    Selection.Clear
    'Add the Axis names
    'copy the costs column names
    Sheets(Sh_from).Select
    Range("BQ" & CStr(1) & ":" & "GL" & CStr(1)).Select
    Selection.Copy
    'paste them
    Sheets(Sh_to).Select
    Cells(1, 1).Select
    Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _
        False, Transpose:=True
    'write them at the right position
    RowSize = z_RowSize(1, Sh_to)
    Col = 2
    For Row = 1 To RowSize Step 6
        Cells(Row + 5, 1).Select
        Selection.Cut
        Cells(1, Col).Select
        ActiveSheet.Paste
        Col = Col + 1
    Next Row
    Columns(1).Select
    Selection.ClearContents
    'Add row names
    Dim Yr As Integer
    Yr = 10
    For iRow = 0 To NrOfRows
        For Row = 2 + 6 * iRow To 7 + 6 * iRow
            Cells(Row, 1).Value = "20" & CStr(Yr)
            Yr = Yr + 1
        Next Row
        Cells(2 + 5, 1).Value = "Total"
    Next iRow
    'Add additional columns
    Columns("A:A").Select
    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
    Columns("A:A").Select
    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
    Range("A1") = "PIdentifier"
    Range("B1") = "ActivityIdentifier"
    Range("C1") = "Year"
    'Format col 1
    Cells.Select
    Selection.HorizontalAlignment = xlCenter

```

```

Selection.ColumnWidth = 10
'***Format row 1
'-----
Range("A1:X1").Select
Rows("1:1").RowHeight = 100
'change orientation
With Selection
    .HorizontalAlignment = xlCenter
    .VerticalAlignment = xlBottom
    .WrapText = True
    .Orientation = 45
    .AddIndent = False
    .IndentLevel = 0
    .ShrinkToFit = False
    .ReadingOrder = xlContext
    .MergeCells = False
End With
'change the color
Range("D1:J1").Select
With Selection.Interior
    .Pattern = xlSolid
    .PatternColorIndex = xlAutomatic
    .ThemeColor = xlThemeColorAccent3
    .TintAndShade = 0.599993896298105
    .PatternTintAndShade = 0
End With
'change the color
Range("K1:Q1").Select
With Selection.Interior
    .Pattern = xlSolid
    .PatternColorIndex = xlAutomatic
    .ThemeColor = xlThemeColorLight2
    .TintAndShade = 0.599993896298105
    .PatternTintAndShade = 0
End With
'change the color
Range("R1:X1").Select
With Selection.Interior
    .Pattern = xlSolid
    .PatternColorIndex = xlAutomatic
    .ThemeColor = xlThemeColorAccent6
    .TintAndShade = 0.399975585192419
    .PatternTintAndShade = 0
End With
Columns("A:A").Select
Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
End Function
Sub test14()
'Create the new spreadsheet "Log_CostTest"
If 1 Then
    Call z_ShNew("Log_CostTest", "Begin")
End If
'Test the cost columns (Works only on the defined Layout)

```

```

If 1 Then
Dim Row_To As Long
Dim Row_From As Long
Dim NrOfRows As Long
'Do some formatting
NrOfRows = 4
    Call z_CostTestFormatting("RD_MasterDataSet7", "Log_CostTest", Row_From, NrOfRows)
Row_From = 2
Row_To = 2 + 6 * 0
    Call z_CostsTest("RD_MasterDataSet7", "Log_CostTest", Row_From, Row_To)
Row_From = 3
Row_To = 2 + 6 * 1
    Call z_CostsTest("RD_MasterDataSet7", "Log_CostTest", Row_From, Row_To)
End If
End Sub

```

```

Sub z_SmCExtractMap_CropProtectionToCropSplit(Sh_from As String, Sh_to As String, Sh_log As String)

```

```

'Make the spreadsheet Flat Value active
'Place the spreadsheet NEW_2011_PI Split by Crop into the same workbook
'Duration 56:00 with ScreenUpdating=True

```

```

'Start the clock
Dim Start, Duration As Date
Start = Now()

```

```

'Remove the blanks " " from the spreadsheet Sh_From
Sheets(Sh_from).Activate
Columns("AB:AJ").Select
ActiveCell.Replace What:=" ", Replacement:="", LookAt:=xlWhole, _
    SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
    ReplaceFormat:=False

```

```

'Enter new columns into the spreadsheet Sh_To
If 1 Then
Sheets(Sh_to).Activate
Rows("1:1").Find(What:="EacFullCosts", LookAt:=xlWhole).Offset(0, 1).Select
Range(ActiveCell, ActiveCell(1, 18)).EntireColumn.Insert
ActiveCell(1, 1).Value = "EAC Full Cost Cereals"
ActiveCell(1, 2).Value = "EAC Full Cost Corn"
ActiveCell(1, 3).Value = "EAC Full Cost DFC"
ActiveCell(1, 4).Value = "EAC Full Cost Rice"
ActiveCell(1, 5).Value = "EAC Full Cost Soybean"
ActiveCell(1, 6).Value = "EAC Full Cost Speciality"
ActiveCell(1, 7).Value = "EAC Full Cost Sugarcane"
ActiveCell(1, 8).Value = "EAC Full Cost Vegetables"
ActiveCell(1, 9).Value = "EAC Full Cost non-Crop"
ActiveCell(1, 10).Value = "EAC Full Cost 2012 Cereals"
ActiveCell(1, 11).Value = "EAC Full Cost 2012 Corn"
ActiveCell(1, 12).Value = "EAC Full Cost 2012 DFC"
ActiveCell(1, 13).Value = "EAC Full Cost 2012 Rice"
ActiveCell(1, 14).Value = "EAC Full Cost 2012 Soybean"

```

```

ActiveCell(1, 15).Value = "EAC Full Cost 2012 Speciality"
ActiveCell(1, 16).Value = "EAC Full Cost 2012 Sugarcane"
ActiveCell(1, 17).Value = "EAC Full Cost 2012 Vegetables"
ActiveCell(1, 18).Value = "EAC Full Cost 2012 non-Crop"
End If

```

```

'Perform the calculations
Application.ScreenUpdating = False
Dim rPIID
Dim ilog As Long: ilog = 2
'Find column indices
Dim Col_TotalFullCosts As Integer
Dim RowSize As Long
Dim Col_PiLevel1 As Integer
Dim Offset_TotalFullCosts_PiLevel1 As Integer
Dim Offset_TotalFullCosts_Pild As Integer
Col_PiLevel1 = z_GetColumnIndex("SyngentaPortfolioLevel1", 1, Sh_to)
Col_TotalFullCosts = z_GetColumnIndex("EacFullCosts", 1, Sh_to)
Col_Pild = z_GetColumnIndex("Pildentifier", 1, Sh_to)
Offset_TotalFullCosts_PiLevel1 = Col_TotalFullCosts - Col_PiLevel1
Offset_TotalFullCosts_Pild = Col_TotalFullCosts - Col_Pild

```

```

'Activate sheet and select range (Look out: some functions like z_GetColumnIndex reselect. Call them
above the cell selection)
'old version: Range(ActiveCell.Offset(0, -24).End(xlDown).Offset(0, 24), ActiveCell(2, 1)).Select
RowSize = z_RowSize(1, Sh_to)
Cells(1, Col_TotalFullCosts).Select
Range(Cells(2, Col_TotalFullCosts), Cells(RowSize, Col_TotalFullCosts)).Select

```

```

'rCropSp: Values of the EACFullCosts column in Sh_To
'rPIID: Values of the Pildentifier column in Sh_To
Dim i As Long: i = 1
For Each rCropSp In Selection.Cells
    rPIID = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild).Value 'Column
    i = i + 1
    If (i Mod 100) = 0 Then
        Debug.Print i
    End If
    'only take those PIs with PortfolioLevel1="CROP_PROTECTION"
    If rCropSp.Offset(0, -Offset_TotalFullCosts_PiLevel1).Value = "CROP_PROTECTION" Then

```

```

        'remove rounding errors to speed up the macro
        rCropSp_flag = "NotNull"
        If rCropSp > -0.001 And rCropSp < 0.001 Then
            rCropSp_flag = "Null"
        End If
        'take only those with costs
        If Not rCropSp_flag = "Null" Then
            rPIID = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild).Value 'Column A
            If Not Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID, LookAt:=xlWhole) Is Nothing
Then
                rCropSp.Offset(0, 1).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 27).Value * rCropSp.Value

```

```

        rCropSp.Offset(0, 2).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 28).Value * rCropSp.Value
        rCropSp.Offset(0, 3).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 29).Value * rCropSp.Value
        rCropSp.Offset(0, 4).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 30).Value * rCropSp.Value
        rCropSp.Offset(0, 5).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 31).Value * rCropSp.Value
        rCropSp.Offset(0, 6).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 32).Value * rCropSp.Value
        rCropSp.Offset(0, 7).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 33).Value * rCropSp.Value
        rCropSp.Offset(0, 8).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 34).Value * rCropSp.Value
        rCropSp.Offset(0, 9).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 35).Value * rCropSp.Value
        'Application.ScreenUpdating = False
        rCropSp.Offset(0, 10).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 27).Value * rCropSp.Offset(0, -2).Value
        rCropSp.Offset(0, 11).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 28).Value * rCropSp.Offset(0, -2).Value
        rCropSp.Offset(0, 12).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 29).Value * rCropSp.Offset(0, -2).Value
        rCropSp.Offset(0, 13).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 30).Value * rCropSp.Offset(0, -2).Value
        rCropSp.Offset(0, 14).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 31).Value * rCropSp.Offset(0, -2).Value
        rCropSp.Offset(0, 15).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 32).Value * rCropSp.Offset(0, -2).Value
        rCropSp.Offset(0, 16).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 33).Value * rCropSp.Offset(0, -2).Value
        rCropSp.Offset(0, 17).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 34).Value * rCropSp.Offset(0, -2).Value
        rCropSp.Offset(0, 18).Value = Sheets(Sh_from).Columns(Col_Pild).Find(What:=rPIID,
LookAt:=xlWhole).Offset(0, 35).Value * rCropSp.Offset(0, -2).Value
    Else
        Sheets(Sh_log).Cells(ilog, 1) = "N/A_in_Sh_From"
        Sheets(Sh_log).Cells(ilog, 2) = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild).Value 'Column
A
        Sheets(Sh_log).Cells(ilog, 3) = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild + 1).Value
'Column B
        Sheets(Sh_log).Cells(ilog, 4) = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild + 2).Value
'Column C
        Sheets(Sh_log).Cells(ilog, 5) = rCropSp
        ilog = ilog + 1
    End If
Else
    Sheets(Sh_log).Cells(ilog, 1) = "EACFullCosts = 0"
    Sheets(Sh_log).Cells(ilog, 2) = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild).Value 'Column A
    Sheets(Sh_log).Cells(ilog, 3) = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild + 1).Value 'Column
B
    Sheets(Sh_log).Cells(ilog, 4) = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild + 2).Value 'Column
C

```



```

        Sheets(Sh_log).Cells(ilog, 5) = rCropSp
        ilog = ilog + 1
    End If
Else
    Sheets(Sh_log).Cells(ilog, 1) = "PortfolioLevel1 <> CP"
    Sheets(Sh_log).Cells(ilog, 2) = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild).Value 'Column A
    Sheets(Sh_log).Cells(ilog, 3) = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild + 1).Value 'Column B
    Sheets(Sh_log).Cells(ilog, 4) = rCropSp.Offset(0, -Offset_TotalFullCosts_Pild + 2).Value 'Column C
    Sheets(Sh_log).Cells(ilog, 5) = rCropSp
    ilog = ilog + 1
End If
Next
Application.ScreenUpdating = True

'Sh_log
Sheets(Sh_log).Activate
Range("A1") = "SkippedBecause"
Range("B1") = "Pildentifier"
Range("C1") = "ActivityIdentifier"
Range("D1") = "PortfolioLevel1"
Range("E1") = "EACFullCosts"

'Stop the clock
Duration = Now() - Start
Sheets(Sh_log).Cells(ilog, 1) = Duration

End Sub

Sub test()
    Call z_SmCMergeExtractWithCropSplitInfo("NEW_2011_PI Split by Crop", "RD_MasterDataSet8")
End Sub

Sub DataCheckForErrorsAndViolatedPreconditions()
    'Checks the data source for errors and whether preconditions of other programs are violated
    'before those are run.
    'Positions with Criteria/conditions that are not satisfied are:
    ' written in a separate spreadsheet called ErrorsAndViolatedPreconditions.
    ' corrected in the source file

    'Variables
    Dim SourceSheet As String: SourceSheet = "Flat Values"
    ActiveWorkbook.Worksheets(SourceSheet).Cells(1, 1).Activate

    Dim RowSize As Variant: RowSize = If(IsEmpty(Range("A1048576")),
    Range("A1048576").End(xlUp).Row, 1048576)

    'Check whether all columns are those expected

    'Check Deleted PIs:
    'ReDim ProjLevel(RowSize) As String

```

```

'Dim i As Integer: i = 0
'ProjLevelDeleted(i) = Left(Cells(i, 2), 2)

'Check Customer Allocation: must be 100 or 0

'Check Portfolio level 1 to 3: compare with online extract

'Check Confidential PIs: If yes is set in column Confidential PIs
'than write Confidential&Right(Identifier,5) in columns PITitle and PILabel
'(and? in PIComment, PIScope, SyngentaProgram)

```

```

'If it does not exist create a separate file for each test
'For i = 1 To Sheets.Count
'  Sheets(i).Name
'Next

```

End Function

Sub AnzeigeMappenNamen()

```

  Dim Wb As Workbook

  For Each Wb In Application.Workbooks
    MsgBox Wb.Name
  Next
End Sub

```

```

Sub AnzeigePfad()
Dim Wb As Workbook
Workbooks(1).Activate
MsgBox ActiveWorkbook.Name
Set Wb = GetObject("C:\Users\t740698\Desktop\RD.xlsm")
Wb.Activate
MsgBox ActiveWorkbook.Name
End Sub

```

```

Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet

```

```

    Sheets(Sh).Activate
    'Determine the row size
    z_RowSize = If(IsEmpty(Cells(1048576, SearchCol)), Cells(1048576, SearchCol).End(xlUp).Row,
1048576)
End Function

```

```

Function z_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As
Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Row, Output: column with the last entry in that row
'SearchCol datatype changed from integer
'Activate the Sheet
Sheets(Sh).Activate
'Determine the col size
z_ColSize = If(IsEmpty(Cells(SearchRow, 16384)), Cells(SearchRow, 16384).End(xlToLeft).Column,
16384)
End Function

```

```

Function z_LastWrittenRowAndCol(Optional Sh As String, Optional StopAtRow As Long, Optional
StopAtCol As Long, Optional ByRef Wb As Workbook) As Variant
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 11.10.2011
'Input:(sh,StopAtRow) or (sh,StopAtRow,0)
'find ColSize_max from 1 to StopAtRow
'find RowSize_max from 1 to ColSize_max
'Input:(sh, ,StopAtCol) or (sh,0,StopAtRow)
'find RowSize_max from 1 to StopAtCol
'find ColSize_max from 1 to RowSize_max
'Input:(sh) or (sh,0,0)
'find RowSize_max from 1 to 16384
'find ColSize_max from 1 to RowSize_max

```

```

Dim ColSize_max As Long: ColSize_max = 0
Dim ColSize_row As Long: ColSize_row = 0
Dim Row As Long
'Optional input
Dim StopAtRow_tmp As Long: StopAtRow_tmp = StopAtRow
If StopAtRow = 0 Then
    StopAtRow = z_LastWrittenRow(Sh, StopAtCol)
End If
For Row = 1 To StopAtRow
    ColSize_row = z_ColSize(Row, Sh)
    If ColSize_row > ColSize_max Then
        ColSize_max = ColSize_row
    End If
Next Row
If StopAtRow_tmp <> 0 Then
    RowSize_max = z_LastWrittenRow(Sh, ColSize_max)
End If
Dim out(0 To 1) As Variant
out(0) = RowSize_max 'The Last written row if StopAtCol is the last written column
out(1) = ColSize_max 'The last written col if StopAtRow is the last written row

```

```
z_LastWrittenRowAndCol = out
End Function
```

```
Function z_LastWrittenRow(Optional Sh As String, Optional StopAtCol As Long, Optional ByRef Wb As
Workbook) As Long
```

```
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

```
'Date: 11.10.2011
```

```
'Input: All Columns, Output: last written column
```

```
'FirstEmptyCol = z_FirstEmptyCol()
```

```
Dim RowSize_max As Long: RowSize_max = 0
```

```
Dim RowSize_col As Long: RowSize_col = 0
```

```
Dim AllCols As Long: AllCols = 16384
```

```
Dim Col As Long
```

```
'Optional input
```

```
If StopAtCol = 0 Then
```

```
    StopAtCol = AllCols
```

```
End If
```

```
For Col = 1 To StopAtCol
```

```
    RowSize_col = z_RowSize(Col, Sh)
```

```
    If RowSize_col > RowSize_max Then
```

```
        RowSize_max = RowSize_col
```

```
    End If
```

```
Next Col
```

```
z_LastWrittenRow = RowSize_max
```

```
End Function
```

```
Sub test()
```

```
    Dim t As Variant
```

```
    t = z_LastWrittenRowAndCol("Log_CustomerChk", 500)
```

```
    Stop
```

```
End Sub
```

```
Public Function z_GetCellIndices(Sh As String, SearchRow As Integer, SearchString As String, Optional
ByRef Wb As Workbook) As Variant
```

```
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

```
Dim CellIndexStr As String 'In R1C1 Format
```

```
Dim CellIndexArr() As String 'Splited R1C1 Format
```

```
Dim ColIndex As Integer
```

```
Dim RowIndex As Integer
```

```
Dim CellIndices(0 To 1) As Integer
```

```
'Activate the right Wb and Sh
```

```
On Error GoTo OptionalArgument:
```

```
Wb.Activate
```

```
On Error GoTo 0
```

```
Sheets(Sh).Activate
```

```
'find column name
```

```
On Error GoTo NameExpectedNotExistent:
```

```
Rows(SearchRow).Find(What:=CStr(SearchString), LookAt:=xlWhole).Select
```

```
'Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole).Select
```

```
On Error GoTo 0
```

```

'find column index
CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
CellIndexArr = Split(CellIndexArr(0), "R")
RowIndex = CInt(CellIndexArr(1))
CellIndices(0) = RowIndex
CellIndices(1) = ColIndex
'Output
z_GetCellIndices = CellIndices
Exit Function

```

OptionalArgument:  
Resume Next

```

NameExpectedNotExistent:
    CellIndices(0) = 0
    CellIndices(1) = 0
    z_GetCellIndices = CellIndices
End Function

```

```

Public Function z_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, _
    Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Output datatype change from Variant

```

```

Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer

```

```

'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
Sheets(Sh).Activate

```

```

'find column name
On Error GoTo NameExpectedNotExistent:
Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole).Select
On Error GoTo 0
'find column index
CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
'Output
z_GetColumnIndex = ColIndex
Exit Function

```

OptionalArgument:  
Resume Next

```

NameExpectedNotExistent:
    ColIndex = 0

```

z\_GetColumnIndex = ColIndex

End Function

Function z\_CopyInsertRange2(Range\_From As Range, Range\_To As Range, Sh\_from As String, Sh\_to As String)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Sheets(Sh\_from).Select

Range\_From.Copy

Sheets(Sh\_to).Select

Range\_To.Insert

End Function

Function z\_CopyInsertRange(RowLU\_From As Long, ColLU\_From As Long, RowRD\_From As Long, ColRD\_From As Long, Sh\_from As String, \_

RowLU\_To As Long, ColLU\_To As Long, Sh\_to As String)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Sheets(Sh\_from).Select

Range(Cells(RowLU\_From, ColLU\_From), Cells(RowRD\_From, ColRD\_From)).Select

Selection.Copy

Sheets(Sh\_to).Select

Cells(RowLU\_To, ColLU\_To).Insert 'moves the other columns to the right

End Function

Function z\_CopyPasteRange2(Range\_From As Range, Range\_To As Range, Sh\_from As String, Sh\_to As String)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Sheets(Sh\_from).Range\_From.Copy Destination:=Sheets(Sh\_to).Range\_To

End Function

Function z\_CopyPasteRange(RowLU\_From As Long, ColLU\_From As Long, RowRD\_From As Long, ColRD\_From As Long, Sh\_from As String, \_

RowLU\_To As Long, ColLU\_To As Long, Sh\_to As String)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Sheets(Sh\_from).Select

Range(Cells(RowLU\_From, ColLU\_From), Cells(RowRD\_From, ColRD\_From)).Copy \_

Destination:=Sheets(Sh\_to).Cells(RowLU\_To, ColLU\_To)

End Function

Function z\_RangeAddressAsArray(Rng As Range) As Variant

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Range(Cells(a,b),Cells(c,d)), Output: Array(a,b,c,d)

sRngAddress = Rng.Address(ReferenceStyle:=xlR1C1)

Dltrlst = Array("\$", "R", "C", ":", ";")

Rpllst = Array("@", "@", "@", "@")

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

    If Rpllst(k) = Empty Then

        l = Rpllst(0)

    Else

        l = k

    End If

    sRngAddress = Replace(sRngAddress, Dltrlst(k), Rpllst(l))

```

Next k
sRngAddress = Replace(sRngAddress, RplLst(0) & RplLst(0), RplLst(l))
Dim RngAddress As Variant
RngAddress = Split(sRngAddress, RplLst(0))
For i = 1 To 4 Step 1
    RngAddress(i - 1) = RngAddress(i)
Next i
ReDim Preserve RngAddress(1 To 4)
z_RangeAddressAsArray = RngAddress
End Function

```

```

Function z_GeneratePivotTable(Piv_ULCell As Range, Source_Rng As Range, _
    Sh_Source As String, Sh_Pivot As String, Piv_F_R_C_V As Variant) As String
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011

```

'Input parameters (arguments) for the z\_GeneratePivotTable function and its call

```

    'Name of the pivot sheet
    'Sh_Pivot = "TimeToDateByTask"
    'Name of the source sheet
    'Sh_Source = "ActualsByWeek"
    'Determine the range of Sh_Source and create a range object
    'RowU = 1: ColL = 1
    'RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
    'Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
    'Create a range object for the upper left corner of the pivot
    'Piv_sULCell = "B9"
    'Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
    'Determine the pivot fields
    'Piv_F_R_C_V(0) = Array(13, 9, 11, 7, 12, 10) 'filter
    'Piv_F_R_C_V(1) = Array(17, 5) 'row labels
    'Piv_F_R_C_V(2) = Array() 'column labels
    'Piv_F_R_C_V(3) = Array(16) 'values

```

'Creat the strings for the function ActiveWorkbook.PivotCaches.Create() further below

```

Dim sSource_Rng As String
sSource_Rng = Sh_Source & "!" & Source_Rng.Address(ReferenceStyle:=xlR1C1)
Dim sPivot_Rng As String
sPivot_Rng = Sh_Pivot & "!" & Piv_ULCell.Address(ReferenceStyle:=xlR1C1)

```

'Store the range.address information into an array

```

Dim RngAddress As Variant
RngAddress = z_RangeAddressAsArray(Source_Rng)

```

'Store the Column names into an array

```

ReDim PivChosenField(RngAddress(2) To RngAddress(4)) As String
For i = RngAddress(2) To RngAddress(4)
    PivChosenField(i) = Source_Rng.Cells(1, i)
Next i

```

'Store the column names of the ReportFilter, RowLabel, ColLabel and Value into arrays

```

ReDim PivReportFilter(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivRowLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivColLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String

```

```

ReDim PivValue(RngAddress(2) - 1 To RngAddress(4) - 1) As String
For i = RngAddress(2) - 1 To RngAddress(4) - 1
    On Error Resume Next
    PivReportFilter(i) = PivChosenField(Piv_F_R_C_V(0)(i))
    On Error GoTo 0
    On Error Resume Next
    PivRowLabel(i) = PivChosenField(Piv_F_R_C_V(1)(i))
    On Error GoTo 0
    On Error Resume Next
    PivCollLabel(i) = PivChosenField(Piv_F_R_C_V(2)(i))
    On Error GoTo 0
    On Error Resume Next
    PivValue(i) = PivChosenField(Piv_F_R_C_V(3)(i))
    On Error GoTo 0
Next i

'generate Pivot
Sheets(Sh_Pivot).Select
Piv_ULCell.Select
ActiveWorkbook.PivotCaches.Create( _
    SourceType:=xlDatabase, _
    SourceData:=sSource_Rng, _
    Version:=xlPivotTableVersion12).CreatePivotTable _
    TableDestination:=sPivot_Rng, _
    TableName:=Piv_Name, _
    DefaultVersion:=xlPivotTableVersion12

'get Pivot table name
'if PivName = "PivotTable" is used more than once an iteger is added to the name
PivName = ActiveSheet.PivotTables(1).Name

For i = RngAddress(2) - 1 To RngAddress(4) - 1
    'Define row labels
    On Error Resume Next
    With ActiveSheet.PivotTables(PivName).PivotFields(PivRowLabel(i))
        .Orientation = xlRowField
        .Position = 1
    End With
    On Error GoTo 0
    'Define column labels
    On Error Resume Next
    With ActiveSheet.PivotTables(PivName).PivotFields(PivCollLabel(i))
        .Orientation = xlColumnField
        .Position = 1
    End With
    On Error GoTo 0
    'Define values
    On Error Resume Next
    ActiveSheet.PivotTables(PivName).AddDataField ActiveSheet.PivotTables( _
    PivName).PivotFields(PivValue(i)), "Sum of STAFF_DAYS", xlSum
    On Error GoTo 0
    'Define report filters
    On Error Resume Next

```



```

With ActiveSheet.PivotTables(PivName).PivotFields(PivReportFilter(i))
    .Orientation = xlPageField
    .Position = 1
End With
On Error GoTo 0
Next i

'Change the layout
With ActiveSheet.PivotTables(PivName)
    .InGridDropZones = True
    .RowAxisLayout xlTabularRow
End With

'Define all columns from : Choose fields to add to report
For i = RngAddress(2) To RngAddress(4)
    ActiveSheet.PivotTables(PivName).PivotFields(PivChosenField(i)).Subtotals = _
    Array(False, False, False, False, False, False, False, False, False, False, False)
Next i

'output
z_GeneratePivotTable = PivName

```

End Function

```

Function z_ThisYear(dDate As Date) As Variant
Dim Yr As Variant
Yr = Year(dDate)
z_ThisYear = Yr
End Function

```

```

Sub test16()
Dim dDate As Date
Dim sOut As String
Dim iOut As Integer
dDate = Date
sOut = z_ThisYear(dDate)
iOut = z_ThisYear(dDate)
End Sub

```

```

'Option Explicit
Public Enum Enum_Col
A_ = 1: B_ = 2: C_ = 3: D_ = 4: E_ = 5: F_ = 6: G_ = 7: H_ = 8: I_ = 9: J_ = 10: K_ = 11: L_ = 12: M_ = 13:
N_ = 14: O_ = 15: P_ = 16: Q_ = 17: R_ = 18: S_ = 19: T_ = 20: U_ = 21: V_ = 22: W_ = 23: X_ = 24: Y_ =
25: z_ = 26
AA_ = 27: AB_ = 28: AC_ = 29: AD_ = 30: AE_ = 31: AF_ = 32: AG_ = 33: AH_ = 34: AI_ = 35: AJ_ = 36:
AK_ = 37: AL_ = 38: AM_ = 39: AN_ = 40: AO_ = 41: AP_ = 42: AQ_ = 43: AR_ = 44: AS_ = 45: AT_ = 46:
AU_ = 47: AV_ = 48: AW_ = 49: AX_ = 50: AY_ = 51: AZ_ = 52
BA_ = 53: BB_ = 54: BC_ = 55: BD_ = 56: BE_ = 57: BF_ = 58: BG_ = 59: BH_ = 60: BI_ = 61: BJ_ = 62:
BK_ = 63: BL_ = 64: BM_ = 65: BN_ = 66: BO_ = 67: BP_ = 68: BQ_ = 69: BR_ = 70: BS_ = 71: BT_ = 72:
BU_ = 73: BV_ = 74: BW_ = 75: BX_ = 76: BY_ = 77: BZ_ = 78

```

```

CA_ = 79: CB_ = 80: CC_ = 81: CD_ = 82: CE_ = 83: CF_ = 84: CG_ = 85: CH_ = 86: CI_ = 87: CJ_ = 88:
CK_ = 89: CL_ = 90: CM_ = 91: CN_ = 92: CO_ = 93: CP_ = 94: CQ_ = 95: CR_ = 96: CS_ = 97: CT_ = 98:
CU_ = 99: CV_ = 100: CW_ = 101: CX_ = 102: CY_ = 103: CZ_ = 104
DA_ = 105: DB_ = 106: DC_ = 107: DD_ = 108: DE_ = 109: DF_ = 110: DG_ = 111: DH_ = 112: DI_ = 113:
DJ_ = 114: DK_ = 115: DL_ = 116: DM_ = 117: DN_ = 118: DO_ = 119: DP_ = 120: DQ_ = 121: DR_ =
122: DS_ = 123: DT_ = 124: DU_ = 125: DV_ = 126: DW_ = 127: DX_ = 128: DY_ = 129: DZ_ = 130
EA_ = 131: EB_ = 132: EC_ = 133: ED_ = 134: EE_ = 135: EF_ = 136: EG_ = 137: EH_ = 138: EI_ = 139:
EJ_ = 140: EK_ = 141: EL_ = 142: EM_ = 143: EN_ = 144: EO_ = 145: EP_ = 146: EQ_ = 147: ER_ = 148:
ES_ = 149: ET_ = 150: EU_ = 151: EV_ = 152: EW_ = 153: EX_ = 154: EY_ = 155: EZ_ = 156
FA_ = 157: FB_ = 158: FC_ = 159: FD_ = 160: FE_ = 161: FF_ = 162: FG_ = 163: FH_ = 164: FI_ = 165:
FJ_ = 166: FK_ = 167: FL_ = 168: FM_ = 169: FN_ = 170: FO_ = 171: FP_ = 172: FQ_ = 173: FR_ = 174:
FS_ = 175: FT_ = 176: FU_ = 177: FV_ = 178: FW_ = 179: FX_ = 180: FY_ = 181: FZ_ = 182
GA_ = 183: GB_ = 184: GC_ = 185: GD_ = 186: GE_ = 187: GF_ = 188: GG_ = 189: GH_ = 190: GI_ =
191: GJ_ = 192: GK_ = 193: GL_ = 194
End Enum

```

```

Sub PrepareTheStringsForTheArrays_ForTheLayoutCheck()

```

```

'Author: Roland Benz, Project Management Excellence

```

```

'Date: 19.9.2011

```

```

'Input: "ExpectedLayout"

```

```

'Output: "ExpectedLayoutToCopyIntoVBA"

```

```

'Objective:

```

```

    'Prepare the input into the array "LayoutExp" in the macro "LayoutCheck" below.

```

```

    'The output of this macro stored in the sheet "ExpectedLayoutToCopyIntoVBA" must

```

```

    'be copied into the macro "LayoutCheck" by the user!(user interaction needed)

```

```

'Create a new spreadsheet

```

```

If 1 Then 'flag: 1=on, 0=off

```

```

On Error Resume Next

```

```

WorksheetExists1 = (Sheets("ExpectedLayoutToCopyIntoVBA").Name <> "")

```

```

On Error GoTo 0

```

```

If WorksheetExists1 = False Then

```

```

    Worksheets.Add(After:=Worksheets(Worksheets.Count)).Name =

```

```

    "ExpectedLayoutToCopyIntoVBA"

```

```

End If

```

```

End If 'flag: 1=on, 0=off

```

```

'ActivityEAC_FV

```

```

If 1 Then

```

```

i = 1

```

```

j = 1

```

```

For k = 3 To 180

```

```

    For l = 1 To 2

```

```

        Sheets("ExpectedLayoutToCopyIntoVBA").Cells(i, j) = Sheets("ExpectedLayout").Cells(k, l)

```

```

        j = j + 1

```

```

        If j Mod 31 = 0 Then

```

```

            j = 1

```

```

            i = i + 1

```

```

        End If

```

```

    Next l

```

```

Next k

```

```

End If

```

```

'PlsEAC_FV

```

```

If 1 Then
i = i + 2
j = 1
For k = 3 To 180
    For l = 3 To 4
        Sheets("ExpectedLayoutToCopyIntoVBA").Cells(i, j) = Sheets("ExpectedLayout").Cells(k, l)
        j = j + 1
        If j Mod 31 = 0 Then
            j = 1
            i = i + 1
        End If
    Next l
Next k
End If
'ResourceReport_FV
If 1 Then
i = i + 2
j = 1
For k = 3 To 180
    For l = 5 To 6
        Sheets("ExpectedLayoutToCopyIntoVBA").Cells(i, j) = Sheets("ExpectedLayout").Cells(k, l)
        j = j + 1
        If j Mod 31 = 0 Then
            j = 1
            i = i + 1
        End If
    Next l
Next k
End If

'Change the content of the cells
If 1 Then
For i = 1 To 40
    For j = 1 To 29 Step 2
        Sheets("ExpectedLayoutToCopyIntoVBA").Cells(i, j) = Chr(34) & CStr(Cells(i, j)) & Chr(34) &
Chr(44) 'Chr(34)=" Chr(44)=,
    Next j
Next i
For i = 1 To 40
    For j = 2 To 30 Step 2
        If j = 30 Then
            Sheets("ExpectedLayoutToCopyIntoVBA").Cells(i, j) = CStr(Cells(i, j)) & Chr(95) & Chr(44) &
Chr(32) & Chr(95) 'Chr(44)=, Chr(32)=Space, Chr(95)=_
        Else
            Sheets("ExpectedLayoutToCopyIntoVBA").Cells(i, j) = CStr(Cells(i, j)) & Chr(95) & Chr(44)
'Chr(34)=" Chr(44)=,
        End If
    Next j
Next i
End If

End Sub

```

```

Sub PrepareTheStringsForTheArrays_ForTheLayoutCheck2()
'Author: Roland Benz, Project Management Excellence
'Date: 19.9.2011
'Input: "MasterLayout" in the workbook CompilationOfMacros!!
'Output: "MasterLayoutToCopyIntoVBA" in the workbook CompilationOfMacros!!
'Objective:
    'Prepare the input into the array "LayoutExp" in the macro "CreateSheetWithWishedLayout"
    below.
    'The output of this macro stored in the sheet "MasterLayoutToCopyIntoVBA" must
    'be copied into the macro "CreateSheetWithWishedLayout" by the user!(user interaction needed)
'Create a new spreadsheet
If 1 Then 'flag: 1=on, 0=off
On Error Resume Next
WorksheetExists1 = (Sheets("MasterLayoutToCopyIntoVBA").Name <> "")
On Error GoTo 0
If WorksheetExists1 = False Then
    Worksheets.Add(After:=Worksheets(Worksheets.Count)).Name = "MasterLayoutToCopyIntoVBA"
End If
End If 'flag: 1=on, 0=off

'RD_MasterDataSet
If 1 Then
i = 1
j = 1
For k = 3 To 200
    For l = 1 To 2
        Sheets("MasterLayoutToCopyIntoVBA").Cells(i, j) = Sheets("MasterLayout").Cells(k, l)
        j = j + 1
        If j Mod 31 = 0 Then
            j = 1
            i = i + 1
        End If
    Next l
Next k
End If
'Change the content of the cells
If 1 Then
For i = 1 To 40
    For j = 1 To 29 Step 2
        Sheets("MasterLayoutToCopyIntoVBA").Cells(i, j) = Chr(34) & CStr(Cells(i, j)) & Chr(34) & Chr(44)
'Chr(34)=" Chr(44)=,
    Next j
Next i
For i = 1 To 40
    For j = 2 To 30 Step 2
        If j = 30 Then
            Sheets("MasterLayoutToCopyIntoVBA").Cells(i, j) = CStr(Cells(i, j)) & Chr(95) & Chr(44) &
Chr(32) & Chr(95) 'Chr(44)=, Chr(32)=Space, Chr(95)=_
        Else
            Sheets("MasterLayoutToCopyIntoVBA").Cells(i, j) = CStr(Cells(i, j)) & Chr(95) & Chr(44)
'Chr(34)=" Chr(44)=,

```

```

        End If
    Next j
Next i
End If
End Sub

```

```

Function z_ChkColExistence(Sh As String, ByRef ColNames(), Sh_log As String) As Boolean
    'The column existence check is assumed to find all column names at the beginning
    z_ChkColExistence = True
    'Determine the column indizes of the ColNames array in Sh
    Dim ColName_i As String
    ReDim Matrix_ColNameColIndex(0 To UBound(ColNames), 0 To 1) As Variant
    'iterate through the array
    For i = LBound(ColNames_from) To UBound(ColNames) Step 1
        ColName_i = CStr(ColNames(i))
        Matrix_ColNameColIndex(i, 0) = ColName_i
        Matrix_ColNameColIndex(i, 1) = z_GetColumnIndex(ColName_i, 1, Sh)
        Debug.Print Matrix_ColNameColIndex(i, 0) & " " & Matrix_ColNameColIndex(i, 1)
        If Matrix_ColNameColIndex(i, 1) = 0 Then
            'write errors into the logfile
            Sheets(Sh_log).Cells(iLog, 2) = "ColName: "
            Sheets(Sh_log).Cells(iLog, 3) = Matrix_ColNameColIndex(i, 0)
            Sheets(Sh_log).Cells(iLog, 4) = "not found in " & Sh
            iLog = iLog + 1
            'The column existence check has detected an unfound column name
            z_ChkColExistence = False
        End If
    Next i
End Function

```

```

Function MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from As String, ByRef
ColNames_from() As Variant, _
    Sh_to As String, ByRef ColNames_to() As Variant) As Variant
    'Determine the column indizes in Sh and Sh_new,
    Dim ColName_from_i As String
    Dim ColName_to_i As String
    ReDim Matrix_ColName1Index1_ColName2Index2(0 To UBound(ColNames_from), 0 To 3) As
Variant

    For i = LBound(ColNames_from) To UBound(ColNames_from) Step 1
        ColName_from_i = CStr(ColNames_from(i))
        Matrix_ColName1Index1_ColName2Index2(i, 0) = ColName_from_i
        Matrix_ColName1Index1_ColName2Index2(i, 1) = z_GetColumnIndex(ColNames_from_i, 1,
Sh_from)
        ColName_to_i = CStr(ColNames_to(i))
        Matrix_ColName1Index1_ColName2Index2(i, 2) = ColName_to_i
        Matrix_ColName1Index1_ColName2Index2(i, 3) = z_GetColumnIndex(ColNames_from_i, 1,
Sh_to)
        Debug.Print Matrix_ColName1Index1_ColName2Index2(i, 0) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 1) _
            & " " & Matrix_ColName1Index1_ColName2Index2(i, 2) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 3)
    Next i

```

```

    MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex =
Matrix_ColName1Index1_ColName2Index2
End Function

```

```

Function z_ShMapColumns(Sh_from As String, ColName_Key_from As String, ByRef
ColNames_from() As Variant, _
    Sh_to As String, ColName_Key_to As String, ByRef ColNames_to() As Variant, _
    Sh_log As String, Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Application.ScreenUpdating = False
'Start time measuring
Dim Start As Date: Dim Duration As Date
Start = Now()

'create a matrix with column names and indexes
MapMatrix = MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from,
ColNames_from, Sh_to, ColNames_to)

'Determine the row size in Sh_to
Sheets(Sh_to).Activate
RowSize_To = z_RowSize(1, Sh_to)

'Find the column index of the KeyName in "Sh_from"
ColIndex_Key_from = z_GetColumnIndex(ColName_Key_from, 1, Sh_from)

'Find the column index of the KeyName in "Sh_to"
ColIndex_Key_to = z_GetColumnIndex(ColName_Key_to, 1, Sh_to)

'Select the range in the column Key_to
Range(Cells(2, ColIndex_Key_to), Cells(RowSize_new, ColIndex_Key_to)).Select

'Iterate throught the rows with "rcheck" = PIdentifier
For Each ValueInCol_Key_to In Selection.Cells
    'if ValueInCol_Key_to is found in Sh_from then perform the mapping
    If Not Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to,
LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns with the help of the MapMatrix
        For j = LBound(MapMatrix) To UBound(MapMatrix) Step 1
            'read the indices
            ColIndex_from_j = MapMatrix(j, 1)
            ColIndex_to_j = MapMatrix(j, 3)
            'map
            ValueInCol_Key_to.Offset(0, (ColIndex_to_j) - ColIndex_Key_to).Value = _
                Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to, _
                LookAt:=xlWhole).Offset(0, (ColIndex_from_j) - ColIndex_Key_from)
        Next j
    Else
        'write not found ValueInCol_Key_to in Sh_from into Sh_log
        Sheets(Sh_log).Cells(ilog, 2) = ValueInCol_Key_to
        Sheets(Sh_log).Cells(ilog, 4) = " not found, map them from another source file Sh_from"
        ilog = ilog + 1
    End If
Next

```

```

'In case the mapping has changed the row height
Sheets(Sh_new).Rows.RowHeight = 15
Application.ScreenUpdating = True
'Write the durations into the logfile
Duration = Now() - Start
Sheets(Sh_log).Cells(ilog + 2, 1) = "Duration" & CStr(Duration): ilog = ilog + 1
End Function

```

```

Sub test()
    Dim MyCell As Range
    Set MyCell = Cells(45, 78)
    Dim CellIndices() As Long
    CellIndices = z_CellToIndex(MyCell)
    Dim MyRange As Range
    Set MyRange = Range(Cells(2, 3), Cells(45, 78))
    Dim RangeIndices() As Long
    RangeIndices = z_RangeToIndices(MyRange)
End Sub

```

```

Sub test55()
    Dim MyRange As Range
    Dim T1 As Boolean
    Dim T2 As Boolean
    Set MyRange = Sheets("Sheet2").Range(Cells(1, 2), Cells(10, 6))
    'Set MyRange = Sheets("Sheet2").Range(Cells(1, 2), Cells(10, 2))
    'Set MyRange = Sheets("Sheet2").Cells(1, 2)
    MyRange.Value = "  "
    T1 = z_IsRangeEmpty("Sheet2", MyRange)
    Call z_TrimCells("Sheet2", MyRange)
    T2 = z_IsRangeEmpty("Sheet2", MyRange)
    Stop
End Sub

```

```

Function z_CellToIndex(ByRef Cell_in As Range) As Variant
    'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim CellIndexStr As String 'In R1C1 Format
    Dim CellIndexArr() As String 'Splited R1C1 Format
    Dim ColIndex As Integer
    Dim RowIndex As Integer
    Dim CellIndices(0 To 1) As Long
    'find column index
    CellIndexStr = Cell_in.Address(ReferenceStyle:=xlR1C1)
    CellIndexArr = Split(CellIndexStr, "C")
    ColIndex = CInt(CellIndexArr(1))
    CellIndexArr = Split(CellIndexArr(0), "R")
    RowIndex = CInt(CellIndexArr(1))
    CellIndices(0) = RowIndex
    CellIndices(1) = ColIndex
    'Output
    z_CellToIndex = CellIndices
End Function

```

```

Function z_sCellToIndex(ByRef CellIndexStr As String) As Variant
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer
Dim RowIndex As Integer
Dim CellIndices(0 To 1) As Long
'find column index
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
CellIndexArr = Split(CellIndexArr(0), "R")
RowIndex = CInt(CellIndexArr(1))
CellIndices(0) = RowIndex
CellIndices(1) = ColIndex
'Output
z_sCellToIndex = CellIndices
End Function

```

```

Function z_RangeToIndices(ByRef Rng As Range) As Variant
Dim RangeIndices(0 To 3) As Long

Dim CellsArray() As String
Dim sAddr As String
sAddr = Rng.Address(ReferenceStyle:=xlR1C1)
CellsArray = Split(sAddr, ":")

Dim CellIndicesUL() As Long
CellIndicesUL = z_sCellToIndex(CellsArray(0))

Dim CellIndicesLR() As Long
On Error GoTo RangelsCell
CellIndicesLR = z_sCellToIndex(CellsArray(1))
On Error GoTo 0

RangeIndices(0) = CellIndicesUL(0)
RangeIndices(1) = CellIndicesUL(1)
RangeIndices(2) = CellIndicesLR(0)
RangeIndices(3) = CellIndicesLR(1)

z_RangeToIndices = RangeIndices
Exit Function
RangelsCell:
CellIndicesLR = z_sCellToIndex(CellsArray(0))
Resume Next
End Function

```

```

Function z_IndicesToRange(RowUL As Long, ColUL As Long, RowDR As Long, ColDR As Long) As Range
Dim Rng As Range
Rng = Range(Cells(RowUL, ColUL), Cells(RowDR, ColDR))
End Function

```

```

Function z_TrimCells(Sh As String, Optional ByRef Rng As Range)
Sheets(Sh).Activate
If Rng Is Nothing Then

```



```

Cells.Select
Else
    Dim MyRngIndices() As Long
    MyRngIndices = z_RangeToIndices(Rng)
    For Row = MyRngIndices(0) To MyRngIndices(2)
        For Col = MyRngIndices(1) To MyRngIndices(3)
            With Excel.WorksheetFunction
                Cells(Row, Col) = .Trim(.Clean(Cells(Row, Col)))
            End With
        Next Col
    Next Row
End If
End Function

Function z_IsRangeEmpty(Sh As String, ByRef Rng As Range) As Boolean
    Sheets(Sh).Activate
    Dim MyIsEmpty As Boolean
    Dim MyRngIndices() As Long
    Dim CellValue_ij As Variant
    MyRngIndices = z_RangeToIndices(Rng)
    For Row = MyRngIndices(0) To MyRngIndices(2)
        For Col = MyRngIndices(1) To MyRngIndices(3)
            CellValue_ij = Cells(Row, Col).Value
            MyIsEmpty = VBA.IsEmpty(CellValue_ij)
        Next Col
    Next Row
    z_IsRangeEmpty = MyIsEmpty
End Function

```

## **File: Makros Zusammenstellung**

```

Sub test()
    With Worksheets("Tabelle1")
        .Rows(1).Font.Bold = True
        .Range("a1:d1").Value = _
            Array("Name", "Full Name", "Title", "Installed")
        For i = 1 To AddIns.Count
            .Cells(i + 1, 1) = AddIns(i).Name
            .Cells(i + 1, 2) = AddIns(i).FullName
            .Cells(i + 1, 3) = AddIns(i).Title
            .Cells(i + 1, 4) = AddIns(i).Installed
        Next
        .Range("a1").CurrentRegion.Columns.AutoFit
    End With
End Sub

Sub FormatiereBereich()
    With Worksheets("Tabelle1").Range("F1:H10")
        .Value = 30
        .Font.Bold = True
    End With
End Sub

```

```
.Interior.Color = RGB(255, 255, 0)
End With
End Sub
```

```
Sub MeineEingabe()
    With Workbooks("Mappe1").Worksheets("Tabelle1").Cells(1, 10)
        .Formula = "=SQRT(50)"
        With .Font
            .Name = "Arial"
            .Bold = True
            .Size = 8
        End With
    End With
End Sub
```

```
Sub speichern()
ActiveWorkbook.SaveAs Filename:="E:\Documents and Settings\Roli\Desktop\myfile"
End Sub
```

```
Sub ShowFolderList(folderspec)
    Dim fs, f, f1, fc, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(folderspec)
    Set fc = f.Files
    For Each f1 In fc
        s = s & f1.Name
        s = s & vbCrLf
    Next
    MsgBox s
End Sub
```

```
Sub ShowDriveList()
    Dim fs, d, dc, s, n
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set dc = fs.Drives
    For Each d In dc
        s = s & d.DriveLetter & " - "
        If d.DriveType = Remote Then
            n = d.ShareName
        Else
            n = d.VolumeName
        End If
        s = s & n & vbCrLf
    Next
    MsgBox s
End Sub
```

```
Sub selectPrinter()
Set bln = Assistant.NewBalloon
With bln
    .Heading = "Select a Printer."
    .Labels(1).Text = "Network Printer"
    .Labels(2).Text = "Local Printer"
```

```

.Labels(3).Text = "Local Color Printer"
.BalloonType = msoBalloonTypeButtons
.Mode = msoModeModeless
.Callback = "ProcessPrinter"
.Show
End With
End Sub

```

```

Sub ProcessPrinter(bln As Balloon, lbtn As Long, _
lPriv As Long)
    Assistant.Animation = msoAnimationPrinting
    Select Case lbtn
    Case -1
        ' Insert network printer-specific code.
    Case -2
        ' Insert local printer-specific code.
    Case -3
        ' Insert color printer-specific code.
    End Select
    bln.Close
End Sub

```

```

Sub Myballoon1()
With Assistant.NewBalloon
    .Heading = "Regional Sales Data"
    .Text = "Select your region"
    For i = 1 To 3
        .CheckBoxes(i).Text = "Region " & i
    Next
    .Button = msoButtonSetOkCancel
    .Show
    If .CheckBoxes(1).Checked Then
        'runregion1
    End If
    If .CheckBoxes(2).Checked Then
        'runregion2
    End If
    If .CheckBoxes(3).Checked Then
        'runregion3
    End If
End With

End Sub

```

```

Sub test1()
Set mc = Worksheets("B-S").Cells(1, 1)
a = mc.Address()           ' $A$1
B = mc.Address(RowAbsolute:=False)    ' $A1
c = mc.Address(ReferenceStyle:=xlR1C1) ' R1C1
d = mc.Address(ReferenceStyle:=xlR1C1, _
    RowAbsolute:=False, _
    ColumnAbsolute:=False, _

```

```
RelativeTo:=Worksheets(1).Cells(3, 3))  
End Sub
```

```
Sub test2()  
a = "H4:C14"  
B = Strings.Split(a, ":")  
c = B(0)  
d = Strings.StrReverse(B(1))  
E = Conversion.Val(d)  
f = Strings.StrReverse(E)  
g = WorksheetFunction.IsNumber(f)  
h = Interaction.IIf(g = False, 1, 2)  
End Sub
```

## **File: Worksheet in RDPIC Phase1 Identify New Components Guidance on next steps 20120419**

```
Sub m_BubbleChart_Add()  
Call z_BubbleChart_Generate("Input", "Chart")  
End Sub
```

```
Function z_BubbleChart_Generate(Sh_Data As String, Sh_Chart As String)  
Sheets(Sh_Chart).Activate  
Cells(1, 1).Activate  
Dim myChartObject As Excel.ChartObject  
Dim myChart As Excel.Chart  
Dim mySeriesCollection As Excel.SeriesCollection  
Dim mySeries As Excel.Series  
Dim myPoints As Excel.Points  
Dim myPoint As Excel.Point  
Dim myChartObjectName As String  
  
'delete all existing charts in sh_chart  
Call z_Chart_Delete(Sh_Chart)  
  
'add a chart object in sh_chart  
Set myChartObject = z_Chart_New(Sh_Chart, xlBubble3DEffect)  
Set myChart = myChartObject.Chart  
myChartObjectName = myChartObject.Name  
  
'resize and reposition the shape  
Call z_ChartObject_SizeAndPosition(Sh_Chart, myChartObjectName, 15, 15, 550, 920)  
  
'set source data range  
Dim nofSeries As Integer  
Dim iter_Series As Integer  
Dim RngAdd As String  
Set mySeriesCollection = myChart.SeriesCollection  
FirstRow = 11  
LastRow = z_RowSize(2, Sh_Data)
```

```

RngAdd = "C" & CStr(FirstRow) & ":" & "G" & CStr(LastRow)
myChart.SetSourceData Source:=Sheets(Sh_Data).Range(RngAdd)
nofSeries = mySeriesCollection.Count
For iter_Series = 1 To nofSeries
    If iter_Series = 1 Then
        Set mySeries = mySeriesCollection.Item(iter_Series)
        mySeries.XValues = "=" & Sh_Data & "!E" & CStr(FirstRow) & ":" & "E" & CStr(LastRow)
        mySeries.Values = "=" & Sh_Data & "!C" & CStr(FirstRow) & ":" & "C" & CStr(LastRow)
        mySeries.BubbleSizes = "=" & Sh_Data & "!G" & CStr(FirstRow) & ":" & "G" & CStr(LastRow)
    Else
        Set mySeries = mySeriesCollection.Item(iter_Series)
        mySeries.Delete
    End If
Next

```

```

'chart title
Call z_Chart_ChartTitle_TextAndFormatAndPosition(Sh_Chart, myChartObjectName, _
    "Visualization of ideas for new portfolio components", 10, 180)

```

```

'Set x and y axis
Dim MinScale_EIBV As Integer
Dim MajUnit_EIBV As Integer
Dim MaxScale_EIBV As Integer
Dim MinScale_Prob As Integer
Dim MajUnit_Prob As Integer
Dim MaxScale_Prob As Integer
MinScale_EIBV = Sheets(Sh_Data).Cells(5, 6)
MajUnit_EIBV = Sheets(Sh_Data).Cells(6, 6)
MaxScale_EIBV = Sheets(Sh_Data).Cells(7, 6)
MinScale_Prob = Sheets(Sh_Data).Cells(5, 4)
MajUnit_Prob = Sheets(Sh_Data).Cells(6, 4)
MaxScale_Prob = Sheets(Sh_Data).Cells(7, 4)

```

```

Call z_Chart_Axes_Layouts(Sh_Chart, myChartObjectName, _
    "Expected incremental Business Value ($M)", "Probability of technical success (%)", _
    MinScale_EIBV, MaxScale_EIBV, MajUnit_EIBV, _
    MinScale_Prob, MaxScale_Prob, MajUnit_Prob)

```

```

'reset the plot area size and position (Make chart and axis title visible first)
Call z_Chart_PlotArea_SizeAndPosition(Sh_Chart, myChartObjectName, 50, 50, 450, 650)

```

```

'remove legend
myChart.SetElement (msoElementLegendNone)

```

```

'delete and add new data label of bubbles
Call z_Chart_Points_DeleteLabels(Sh_Chart, myChartObjectName) 'not necessary here
Dim LabelRange As Range
Sheets(Sh_Data).Activate
Set LabelRange = Sheets(Sh_Data).Range(Cells(FirstRow, 2), Cells(LastRow, 2))
Call z_Chart_Points_AddLabels(Sh_Data, Sh_Chart, LabelRange, "Center")

```

```

'Point formatting
Dim AttributeRange As Range

```

```

Sheets(Sh_Data).Activate
Dim DropDownList As Range
Set DropDownList = Range(Cells(5, 9), Cells(10, 9))
Set AttributeRange = Sheets(Sh_Data).Range(Cells(FirstRow, 9), Cells(LastRow, 9))
Call z_Chart_Point_Format(Sh_Data, Sh_Chart, myChartObjectName, DropDownList,
AttributeRange, "IsArea")
'chart style of bubbles
myChart.ChartStyle = 31

'Add textboxes
Call z_Chart_Textboxes_Add(Sh_Data, Sh_Chart, myChartObjectName, _
    MinScale_EIBV, MaxScale_EIBV, _
    MinScale_Prob, MaxScale_Prob)

```

End Function

```

Private Sub m_Chart_Delete()
    Call z_Chart_Delete("Chart2")
End Sub

```

```

Private Sub m_Chart_New()
    Dim myChartObject As Excel.ChartObject
    Set myChartObject = z_Chart_New("Chart2", xlBubble3DEffect)
End Sub

```

```

Private Sub m_ChartObject_SizeAndPosition()
    Call z_ChartObject_SizeAndPosition("Chart", "Chart 1", 15, 15, 550, 900)
    'Call z_ChartObject_SizeAndPosition("Chart", "Chart 1", 15, 15, 350, 900)
End Sub

```

```

Private Sub m_Chart_PlotArea_SizeAndPosition()
    Call z_Chart_PlotArea_SizeAndPosition("Chart", "Chart 1", 50, 50, 450, 600)
End Sub

```

```

Private Sub m_Chart_ChartTitle_TextAndFormatAndPosition()
    Call z_Chart_ChartTitle_TextAndFormatAndPosition("Chart", "Chart 1", "Visualization of ideas for
new portfolio components", 5, 133.5)
End Sub

```

```

Private Sub m_Chart_Axes_Layouts()
    Call z_Chart_Axes_Layouts("Chart", "Chart 1", _
        "Expected incremental Business Value ($M)", "Probability of technical success (%)", _
        0, 250, 50, 0, 100, 20)
End Sub

```

```

Private Sub m_Chart_Points_DeleteLabels()
    Call z_Chart_Points_DeleteLabels("Chart", "Chart 1")
End Sub

```

```

Private Sub m_Chart_Points_AddLabels()
    Dim LabelRange As Range
    Sheets("Input").Activate
    Set LabelRange = Sheets("Input").Range(Cells(11, 2), Cells(30, 2))

```

```

    Call z_Chart_Points_AddLabels("Input", "Chart", LabelRange, "Center")
End Sub

```

```

Private Sub m_Chart_Point_Format()
    Dim AttributeRange As Range
    Sheets("Input").Activate
    Dim DropDownList As Range
    Set DropDownList = Range(Cells(5, 9), Cells(10, 9))
    Set AttributeRange = Sheets("Input").Range(Cells(11, 9), Cells(30, 9))
    Call z_Chart_Point_Format("Input", "Chart", "Chart 1", DropDownList, AttributeRange, "IsArea")
End Sub

```

```

Private Sub m_Chart_Textboxes_Add()
    Call z_Chart_Textboxes_Add("Input", "Chart2", "Chart 2")
End Sub

```

```

Private Sub m_ChartObjects_GetNames()
    Dim ChartName As Variant
    ChartName = z_ChartObjects_GetNames("Chart2")
    ChartName = z_ChartObjects_GetNames("Chart2", "Chart Title 3")
    a = UBound(ChartName) - LBound(ChartName)
End Sub

```

```

Private Sub m_ChartObject_Visibility()
    Sh_Chart = "Chart2"
    Call z_ChartObject_Visibility(Sheets(Sh_Chart).ChartObjects(1), "front")
    Call z_ChartObject_Visibility(Sheets(Sh_Chart).ChartObjects(1), "visible")
End Sub

```

```

Function z_Chart_Delete(Sh_Chart As String, Optional ChartObjectName As String)
    Sheets(Sh_Chart).Activate
    Cells(1, 1).Activate
    Dim myChartObject As Excel.ChartObject
    Dim myChart As Excel.Chart
    Dim nofCharts As Integer
    nofCharts = Sheets(Sh_Chart).ChartObjects.Count
    Dim iter As Integer
    'delete all
    If ChartObjectName = Empty Then
        For iter = 1 To nofCharts
            'always delete the first (last becomes the first after a while)
            If Sheets(Sh_Chart).ChartObjects(1).Visible = False Then
                Sheets(Sh_Chart).ChartObjects(1).Visible = True
                Set myChartObject = Sheets(Sh_Chart).ChartObjects(1)
                Set myChart = myChartObject.Chart
                myChartObject.Activate
                myChartObject.Delete
                'myChart.Delete
            Else
                Set myChartObject = Sheets(Sh_Chart).ChartObjects(1)
                Set myChart = myChartObject.Chart
            End If
        Next iter
    End If
End Function

```

```

        myChartObject.Activate
        myChartObject.Delete
        'myChart.Delete
    End If
Next
'delete ChartObjectName
Else
    For iter = 1 To nofCharts
        If Sheets(Sh_Chart).ChartObjects(iter).Visible = False Then
            Sheets(Sh_Chart).ChartObjects(iter).Visible = True
            Set myChartObject = Sheets(Sh_Chart).ChartObjects(iter)
            Set myChart = myChartObject.Chart
            If myChartObject.Name = ChartObjectName Then
                myChartObject.Activate
                myChartObject.Delete
                'myChart.Delete
            Exit For
        End If
    Else
        Set myChartObject = Sheets(Sh_Chart).ChartObjects(iter)
        Set myChart = myChartObject.Chart
        If myChartObject.Name = ChartObjectName Then
            myChartObject.Activate
            myChartObject.Delete
            'myChart.Delete
        Exit For
    End If
End If
Next
End If
End Function

```

```

Function z_Chart_New(Sh_Chart As String, ChartType As Integer) As Excel.ChartObject
    Dim myChartObject As Excel.ChartObject
    Dim myChart As Excel.Chart
    Sheets(Sh_Chart).Activate
    ActiveSheet.Shapes.AddChart.Select
    Set myChart = ActiveChart
    myChart.ChartType = ChartType
    Set myChartObject = myChart.Parent
    Set z_Chart_New = myChartObject
End Function

```

```

Function z_ChartObject_SizeAndPosition(Sh_Chart As String, ChartObjectName As String, _
    T_ As Integer, L_ As Integer, H_ As Integer, W_ As Integer)
    Sheets(Sh_Chart).Activate
    Cells(1, 1).Activate
    Dim myChartObject As Excel.ChartObject
    Set myChartObject = Sheets(Sh_Chart).ChartObjects(ChartObjectName)
    With myChartObject
        .Top = T_ ' reposition
        .Left = L_ ' reposition
        .Height = H_ ' resize
    End With

```



```

        .Width = W_ ' resize
    End With
End Function

```

```

Function z_Chart_PlotArea_SizeAndPosition(Sh_Chart As String, ChartObjectName As String, _
    Optional T_ As Integer, Optional L_ As Integer, Optional H_ As Integer, Optional W_ As Integer)
    Sheets(Sh_Chart).Activate
    Cells(1, 1).Activate
    Dim myChart As Excel.Chart
    Set myChart = Sheets(Sh_Chart).ChartObjects(ChartObjectName).Chart
    myChart.SetElement msoElementPlotAreaShow
    myChart.PlotArea.Position = xlChartElementPositionAutomatic
    'reset the chart size and position
    If T_ <> Empty Then
        myChart.PlotArea.Top = T_
    End If
    If L_ <> Empty Then
        myChart.PlotArea.Left = L_
    End If
    If H_ <> Empty Then
        myChart.PlotArea.Height = H_
    End If
    If W_ <> Empty Then
        myChart.PlotArea.Width = W_
    End If
End Function

```

```

Function z_Chart_ChartTitle_TextAndFormatAndPosition(Sh_Chart As String, ChartObjectName As
String, _
    TitleText As String, Optional T_ As Integer, Optional L_ As Integer)
    Sheets(Sh_Chart).Activate
    Cells(1, 1).Activate
    Dim myChart As Excel.Chart
    Set myChart = Sheets(Sh_Chart).ChartObjects(ChartObjectName).Chart
    myChart.HasTitle = True
    myChart.ChartTitle.Text = TitleText
    myChart.ChartTitle.Characters.Font.FontStyle = "Calibri"
    myChart.ChartTitle.Characters.Font.Size = 18
    myChart.ChartTitle.Characters.Font.Bold = True
    myChart.ChartTitle.Characters.Font.Color = RGB(0, 0, 0)
    myChart.ChartTitle.Characters.Font.Underline = True
    myChart.SetElement (msoElementChartTitleAboveChart)
    myChart.ChartTitle.Position = xlChartElementPositionAutomatic
    If T_ <> Empty Then
        myChart.ChartTitle.Top = T_
    End If
    If L_ <> Empty Then
        myChart.ChartTitle.Left = L_
    End If
End Function

```

```

Function z_Chart_Axes_Layouts(Sh_Chart As String, ChartObjectName As String, _
    xAxisName As String, yAxisName As String, _

```

```

Optional xMin As Integer, Optional xMax As Integer, Optional xMaj As Integer, _
Optional yMin As Integer, Optional yMax As Integer, Optional yMaj As Integer, _
Optional xCross As Integer, Optional yCross As Integer)
Sheets(Sh_Chart).Activate
Cells(1, 1).Activate
Dim myChart As Excel.Chart
Set myChart = Sheets(Sh_Chart).ChartObjects(ChartObjectName).Chart
myChart.Parent.Activate

'x axis scale
Dim myXaxis As Excel.Axis
myChart.HasAxis(xlCategory) = True
Set myXaxis = myChart.Axes(xlCategory)
myXaxis.Select
myXaxis.MinimumScale = xMin
myXaxis.MaximumScale = xMax
myXaxis.MajorUnit = xMaj
myXaxis.Format.Line.Weight = 2.5

'y axis scale
Dim myYaxis As Excel.Axis
myChart.HasAxis(xlValue) = True
Set myYaxis = myChart.Axes(xlValue)
myYaxis.Select
myYaxis.MinimumScale = yMin
myYaxis.MaximumScale = yMax
myYaxis.MajorUnit = yMaj
myYaxis.Format.Line.Weight = 2.5

'y axis crosses x axis at
If xCross <> Empty Then
    myXaxis.CrossesAt = xCross
Else
    myXaxis.CrossesAt = (xMax - xMin) / 2
End If

'x axis crosses y axis at
If yCross <> Empty Then
    myYaxis.CrossesAt = yCross
Else
    myYaxis.CrossesAt = (yMax - yMin) / 2
End If

'add x axis title
myChart.HasAxis(xlCategory, xlPrimary) = True
myChart.SetElement (msoElementPrimaryCategoryAxisTitleAdjacentToAxis)
myXaxis.AxisTitle.Select
myXaxis.AxisTitle.Text = xAxisName
myXaxis.AxisTitle.Characters.Font.FontStyle = "Calibri"
myXaxis.AxisTitle.Characters.Font.Size = 16
myXaxis.AxisTitle.Characters.Font.Bold = True
myXaxis.AxisTitle.Characters.Font.Color = RGB(0, 0, 0)

```

```

'add y axis title
myChart.HasAxis(xlValue, xlPrimary) = True = True
myChart.SetElement (msoElementPrimaryValueAxisTitleRotated)
myYaxis.AxisTitle.Select
myYaxis.AxisTitle.Text = yAxisName
myYaxis.AxisTitle.Characters.Font.FontStyle = "Calibri"
myYaxis.AxisTitle.Characters.Font.Size = 16
myYaxis.AxisTitle.Characters.Font.Bold = True
myYaxis.AxisTitle.Characters.Font.Color = RGB(0, 0, 0)

'grid on
ActiveChart.SetElement (msoElementPrimaryCategoryGridLinesMajor)
ActiveChart.SetElement (msoElementPrimaryValueGridLinesMajor)
End Function

Sub z_Chart_Points_AddLabels(Sh_Sorce As String, Sh_Chart As String, _
    LabelRange As Range, CenterOrBelowOrBestFit As String)
    Sheets(Sh_Chart).Activate
    Cells(1, 1).Activate
    Dim myChartObject As Excel.ChartObject
    Dim myChart As Excel.Chart
    Dim mySeriesCollection As Excel.SeriesCollection
    Dim mySeries As Excel.Series
    Dim myPoints As Excel.Points
    Dim myPoint As Excel.Point
    Dim Iter_Chart As Integer
    Dim iter_Series As Integer
    Dim iter_Point As Integer
    'iterate through all chartsObjects(and its chart and seriesCollection) in Sh_chart
    Dim nofCharts As Integer
    nofCharts = Sheets(Sh_Chart).ChartObjects.Count
    For Iter_Chart = 1 To nofCharts
        Set myChartObject = Sheets(Sh_Chart).ChartObjects(Iter_Chart)
        Set myChart = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart
        Set mySeriesCollection = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection
        'iterate through all series and change the properties of all points (bubbles) of each series
        Dim nofSeries As Integer
        nofSeries = myChart.SeriesCollection.Count
        For iter_Series = 1 To nofSeries
            Set mySeries = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series)
            mySeries.ApplyDataLabels
            mySeries.DataLabels.ShowValue = True
            'mySeries.DataLabels.Position = xlLabelPositionInsideBase
            mySeries.DataLabels.Font.Name = "Calibri"
            mySeries.DataLabels.Font.Size = 9
            mySeries.DataLabels.Font.Color = RGB(0, 0, 0)
            'iterate through all points (bubbles) of a series and change the properties of the points
            Set myPoints =
            Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series).Points
            nofPoints = myPoints.Count
            For iter_Point = 1 To nofPoints
                Set myPoint =
                Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series).Points(iter_Point)
            
```

```

        myPoint.Select
    On Error Resume Next
    myPoints(iter_Point).HasDataLabel = True
    If CenterOrBelowOrBestFit = "Center" Then
        myPoints(iter_Point).DataLabel.Position = xlLabelPositionCenter
    ElseIf CenterOrBelowOrBestFit = "Below" Then
        myPoints(iter_Point).DataLabel.Position = xlLabelPositionBelow
    ElseIf CenterOrBelowOrBestFit = "BestFit" Then
        myPoints(iter_Point).DataLabel.Position = xlLabelPositionBestFit
    Else
        myPoints(iter_Point).DataLabel.Position = xlLabelPositionBestFit
    End If
    myPoints(iter_Point).DataLabel.Text = LabelRange(iter_Point, 1)
    On Error GoTo 0
Next
Next
Next
End Sub

Function z_Chart_Point_Format(Sh_Sorce As String, Sh_Chart As String, ChartObjectName As String,
-
    Optional DropDownList As Range, Optional AttributeRange As Range, _
    Optional IsWidthOrIsArea As String)
    Sheets(Sh_Chart).Activate
    Cells(1, 1).Activate
    Dim myChartObject As Excel.ChartObject
    Dim myChart As Excel.Chart
    Dim mySeriesCollection As Excel.SeriesCollection
    Dim mySeries As Excel.Series
    Dim myPoints As Excel.Points
    Dim myPoint As Excel.Point
    Dim Iter_Chart As Integer
    Dim Iter_ChartGroup As Integer
    Dim iter_Series As Integer
    Dim iter_Point As Integer
    'iterate through all chartObjects(and its chart and seriesCollection) in Sh_chart
    Dim nofCharts As Integer
    nofCharts = Sheets(Sh_Chart).ChartObjects.Count
    For Iter_Chart = 1 To nofCharts
        Set myChartObject = Sheets(Sh_Chart).ChartObjects(Iter_Chart)
        Set myChart = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart
        Set mySeriesCollection = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection
        If myChartObject.Name = ChartObjectName Then
            Dim nofChartGroups As Integer
            nofChartGroups = myChart.ChartGroups.Count
            For Iter_ChartGroup = 1 To nofChartGroups
                If IsWidthOrIsArea = "IsWidth" Then
                    myChart.ChartGroups(Iter_ChartGroup).SizeRepresents = xlSizelsWidth
                ElseIf IsWidthOrIsArea = "IsArea" Then
                    myChart.ChartGroups(Iter_ChartGroup).SizeRepresents = xlSizelsArea
                Else
                    myChart.ChartGroups(Iter_ChartGroup).SizeRepresents = xlSizelsArea
                End If
            End For
        End If
    End For
End Function

```

```

Next
'iterate through all series and change the properties of all points (bubbles) of each series
Dim nofSeries As Integer
nofSeries = myChart.SeriesCollection.Count
For iter_Series = 1 To nofSeries
    Set mySeries = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series)
    'iterate through all points (bubbles) of a series and change the properties of the points
    Set myPoints =
Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series).Points
    nofPoints = myPoints.Count
    For iter_Point = 1 To nofPoints
        Set myPoint =
Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series).Points(iter_Point)
        myPoint.Select
        On Error Resume Next
        If AttributeRange Is Nothing Then
            'chart style of bubbles
            myChart.ChartStyle = 31
        Else
            'mypoint.ClearFormats
            Dim AttributeValue As String
            AttributeValue = AttributeRange(iter_Point, 1).Value
            If AttributeValue = DropDownList(1, 1) Then
                myPoint.Format.Fill.ForeColor.RGB = RGB(255, 0, 0) 'red
            ElseIf AttributeValue = DropDownList(2, 1) Then
                myPoint.Format.Fill.ForeColor.RGB = RGB(0, 255, 0) 'green
            ElseIf AttributeValue = DropDownList(3, 1) Then
                myPoint.Format.Fill.ForeColor.RGB = RGB(0, 0, 255) 'blue
            ElseIf AttributeValue = DropDownList(4, 1) Then
                myPoint.Format.Fill.ForeColor.RGB = RGB(255, 153, 0) 'orange
            ElseIf AttributeValue = DropDownList(5, 1) Then
                myPoint.Format.Fill.ForeColor.RGB = RGB(204, 0, 255) 'violett
            ElseIf AttributeValue = DropDownList(6, 1) Then
                myPoint.Format.Fill.ForeColor.RGB = RGB(148, 138, 84) 'tan
            Else
                'Stop
            End If
        End If
        myPoint.Format.Fill.Transparency = 0.6
        On Error GoTo 0
    Next
Next
Next
Else
    Stop
End If
Next
End Function

Function z_Chart_Textboxes_Add(Sh_Data As String, Sh_Chart As String, ChartObjectName As String,
-
    Optional xMin As Integer, Optional xMax As Integer, _
    Optional yMin As Integer, Optional yMax As Integer)
    Sheets(Sh_Chart).Activate

```

```

Cells(1, 1).Activate
Dim WSh_Data As Worksheet
Dim WSh_Chart As Worksheet
Dim myChartObject As Excel.ChartObject
Dim myChart As Excel.Chart
Dim myShape1 As Shape
Dim myText1 As String
Dim myShape2 As Shape
Dim myText2 As String

Set WSh_Data = Sheets(Sh_Data)
Set WSh_Chart = Sheets(Sh_Chart)
Set myChartObject = Sheets(Sh_Chart).ChartObjects(ChartObjectName)
Set myChart = myChartObject.Chart

'Delete all textboxes
Shapes_count = myChart.Shapes.Count
myChart.Shapes.SelectAll
For iter_Shapes = Shapes_count To 1 Step -1
    If myChart.Shapes(iter_Shapes).Type = msoTextBox Then
        myChart.Shapes(iter_Shapes).Delete
    End If
Next

'add new textboxes
Dim Idea_h As String
Dim Idea_l As String
Dim Cost_h As Double
Dim Cost_l As Double
Dim FirstRow As Integer
Dim LastRow As Integer
FirstRow = 11
LastRow = z_RowSize(2, Sh_Data)

'Determine the lowest and highest costs
'Worksheet must be activated for range object
WSh_Data.Activate
'unconditional
Cost_h = WorksheetFunction.Max(WSh_Data.Range(Cells(FirstRow, 7), Cells(LastRow, 7)))
Cost_l = WorksheetFunction.Min(WSh_Data.Range(Cells(FirstRow, 7), Cells(LastRow, 7)))
'conditional i.e. in the range of xMin, xMax and yMin, yMax
Dim iter As Integer
Dim ValX As Double
Dim ValY As Double
Dim Cost As Double
Dim Cost_h_cond As Double
Dim Cost_l_cond As Double
Cost_h_cond = Cost_l
Cost_l_cond = Cost_h
For iter = 1 To LastRow - FirstRow + 1
    ValY = WSh_Data.Range(Cells(FirstRow, 3), Cells(LastRow, 3))(iter, 1)
    ValX = WSh_Data.Range(Cells(FirstRow, 5), Cells(LastRow, 5))(iter, 1)
    Cost = WSh_Data.Range(Cells(FirstRow, 7), Cells(LastRow, 7))(iter, 1)

```

```

If ValX >= xMin And ValX <= xMax And ValY >= yMin And ValY <= yMax Then
    If Cost > Cost_h_cond Then
        Cost_h_cond = Cost
    End If
    If Cost < Cost_l_cond Then
        Cost_l_cond = Cost
    End If
End If
Next
Cost_h = Cost_h_cond
Cost_l = Cost_l_cond

'because of merged cells find method only returns a value if the range goes over columns 7 and 8
On Error Resume Next
Idea_h = Cells(WSh_Data.Range(Cells(FirstRow, 7), Cells>LastRow, 8)).Find(What:=Cost_h,
LookAt:=xlWhole).Row, 2).Value
Idea_l = Cells(WSh_Data.Range(Cells(FirstRow, 7), Cells>LastRow, 8)).Find(What:=Cost_l,
LookAt:=xlWhole).Row, 2).Value
On Error GoTo 0

WSh_Chart.Activate
Set myShape1 = myChart.Shapes.AddTextbox(msoTextOrientationHorizontal, _
    700, 55, 180, 140)
myShape1.Fill.BackColor.RGB = RGB(250, 250, 250)
myText1 = "Bubble Size:" & Chr(13) & "Estimated Total Cost ($M)" & Chr(13) & Chr(13) & _
    Idea_h & Chr(13) & "has the highest costs : " & CStr(Cost_h) & Chr(13) & Chr(13) & _
    Idea_l & Chr(13) & "has the lowest costs : " & CStr(Cost_l) & Chr(13)
myShape1.TextFrame.Characters.Text = myText1
myShape1.TextFrame.Characters.Font.FontStyle = "Calibri"
myShape1.TextFrame.Characters.Font.Size = 11
myShape1.TextFrame.Characters(1, 39).Font.Size = 12
myShape1.TextFrame.Characters(1, 39).Font.Bold = True
myShape1.TextFrame.Characters(1, 39).Font.Color = RGB(0, 0, 0)
myShape1.TextFrame.Characters(40, 100).Font.Bold = False
myShape1.TextFrame.Characters(40, 100).Font.Color = RGB(0, 0, 0)

Set myShape2 = myChart.Shapes.AddTextbox(msoTextOrientationHorizontal, _
    700, 220, 180, 200)
myShape2.Fill.BackColor.RGB = RGB(250, 250, 250)
myText2 = "Bubble Color:" & Chr(13) & "Portfolio Component Category" & Chr(13) & Chr(13) & _
    "Integrated Crop Solutions" & Chr(13) & Chr(13) & "New Formulations" & Chr(13) & Chr(13) &
    "Non-Product Customer Offers" & Chr(13) & Chr(13) & "Performance Enhancement Research"
    & Chr(13) & Chr(13) & _
    "Product & Solution Extensions" & Chr(13) & Chr(13) & "Other" & Chr(13)
myShape2.TextFrame.Characters.Text = myText2
myShape2.TextFrame.Characters.Font.FontStyle = "Calibri"
myShape2.TextFrame.Characters.Font.Size = 11
myShape2.TextFrame.Characters(1, 43).Font.Size = 12
myShape2.TextFrame.Characters(1, 43).Font.Bold = True
myShape2.TextFrame.Characters(1, 43).Font.Color = RGB(0, 0, 0)
myShape2.TextFrame.Characters(44, 26).Font.Color = RGB(255, 0, 0) 'red
myShape2.TextFrame.Characters(71, 18).Font.Color = RGB(0, 255, 0) 'green

```

```

myShape2.TextFrame.Characters(89, 30).Font.Color = RGB(0, 0, 255) 'blue
myShape2.TextFrame.Characters(119, 32).Font.Color = RGB(255, 153, 0) 'orange
myShape2.TextFrame.Characters(152, 30).Font.Color = RGB(204, 0, 255) 'violet
myShape2.TextFrame.Characters(182, 8).Font.Color = RGB(148, 138, 84) 'tan

```

```

'myPoint.Format.Fill.Transparency = 60
End Function

```

```

Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As
Long

```

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011

```

```

    z_RowSize = If(IsEmpty(Sheets(Sh).Cells(1048576, SearchCol)), Sheets(Sh).Cells(1048576,
SearchCol).End(xlUp).Row, 1048576)
End Function

```

```

Function z_ChartObjects_GetNames(Sh_Chart As String, Optional ChartTitle As String) As Variant

```

```

    Sheets(Sh_Chart).Activate
    Cells(1, 1).Activate
    Dim myChartObject As Excel.ChartObject
    Dim myChart As Excel.Chart
    Dim nofCharts As Integer
    nofCharts = Sheets(Sh_Chart).ChartObjects.Count
    ReDim ChartObjectNames(0 To nofCharts - 1) As String
    Dim iter As Integer
    'chart title input empty
    If ChartTitle = Empty Then
        'iterate though all chartobjects in the sheet
        'For Each Chart_ In ActiveSheet.ChartObjects
        '    Chart_.Activate
        '    ChartName = Split(Chart_.Name, " ")
        '    ChartName = ChartName(1) & " " & ChartName(2)
        '    ChartObjectName = Chart_.Parent.Name
        'Next
        For iter = 1 To nofCharts
            If Sheets(Sh_Chart).ChartObjects(iter).Visible = False Then
                Sheets(Sh_Chart).ChartObjects(iter).Visible = True
                Set myChartObject = Sheets(Sh_Chart).ChartObjects(iter)
                Set myChart = myChartObject.Chart
                If myChart.HasTitle = False Then
                    myChart.HasTitle = True
                    ChartObjectNames(iter - 1) = myChartObject.Name
                    myChart.HasTitle = False
                Else
                    ChartObjectNames(iter - 1) = myChartObject.Name
                End If
                myChartObject.Visible = False
            Else
                Set myChartObject = Sheets(Sh_Chart).ChartObjects(iter)
                Set myChart = myChartObject.Chart
                myChartObject.Activate
                If myChart.HasTitle = False Then

```



```

        myChart.HasTitle = True
        ChartObjectNames(iter - 1) = myChartObject.Name
        myChart.HasTitle = False
    Else
        ChartObjectNames(iter - 1) = myChartObject.Name
    End If
End If
Next
'chart title input not empty
Else
    For iter = 1 To nofCharts
        If Sheets(Sh_Chart).ChartObjects(iter).Visible = False Then
            Sheets(Sh_Chart).ChartObjects(iter).Visible = True
            Set myChartObject = Sheets(Sh_Chart).ChartObjects(iter)
            Set myChart = myChartObject.Chart
            If myChart.HasTitle = False Then
                myChart.HasTitle = True
                If myChart.ChartTitle.Text = ChartTitle Then
                    ChartObjectNames(0) = myChartObject.Name
                    Exit For
                End If
                myChart.HasTitle = False
            Else
                If myChart.ChartTitle.Text = ChartTitle Then
                    ChartObjectNames(0) = myChartObject.Name
                    Exit For
                End If
            End If
            myChartObject.Visible = False
        Else
            Set myChartObject = Sheets(Sh_Chart).ChartObjects(iter)
            Set myChart = myChartObject.Chart
            myChartObject.Activate
            If myChart.HasTitle = False Then
                myChart.HasTitle = True
                If myChart.ChartTitle.Text = ChartTitle Then
                    ChartObjectNames(0) = myChartObject.Name
                    Exit For
                End If
                myChart.HasTitle = False
            Else
                If myChart.ChartTitle.Text = ChartTitle Then
                    ChartObjectNames(0) = myChartObject.Name
                    Exit For
                End If
            End If
        End If
    Next
End If
z_ChartObjects_GetNames = ChartObjectNames
End Function

Function z_ChartObject_Visibility(myChartObject As Excel.ChartObject, Visibility As String)

```

```

'invisible
If Visibility = "invisible" Then
    myChartObject.Visible = False
'visible
Elseif Visibility = "visible" Then
    myChartObject.Visible = True
'front
Elseif Visibility = "front" Then
    myChartObject.BringToFront
'back
Elseif Visibility = "back" Then
    myChartObject.SendToBack
Else
    Stop
End If
End Function

```

```

Function z_Chart_Points_DeleteLabels(Sh_Chart As String, ChartObjectName As String)
    Sheets(Sh_Chart).Activate
    Dim nofCharts As Integer
    nofCharts = Sheets(Sh_Chart).ChartObjects.Count
    For Iter_Chart = 1 To nofCharts
        Set myChartObject = Sheets(Sh_Chart).ChartObjects(Iter_Chart)
        Set myChart = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart
        Set mySeriesCollection = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection
        If myChartObject.Name = ChartObjectName Then
            'iterate through all series and change the properties of all points (bubbles) of each series
            Dim nofSeries As Integer
            nofSeries = myChart.SeriesCollection.Count
            For iter_Series = 1 To nofSeries
                Set mySeries = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series)
                On Error Resume Next
                mySeries.DataLabels.Delete
                On Error GoTo 0
            Next
        End If
    Next
End Function

```

Option Explicit

```

Private Sub m_AddBubbleChart()
    Call z_Add_BubbleChart("Input", "Chart2")
End Sub
Private Function z_AddBubbleChart(Sh_Data As String, Sh_Chart As String)
    'delet all existing charts in sh_chart

    'add chart object
    Sheets(Sh_Chart).Activate
    ActiveSheet.Shapes.AddChart.Select
    'set chart object
    Dim myChart As Excel.Chart

```

```

Set myChart = ActiveChart
'resize and reposition the shape
With myChart.Parent
    .Height = 550 'resize
    .Width = 900 'resize
    .Top = 15 'reposition
    .Left = 15 'reposition
End With
'change chart type
myChart.ChartType = xlBubble3DEffect
'set source data range
myChart.SetSourceData Source:=Sheets(Sh_Data).Range("C8:G19")
'change the preset assignment of the data to the series 1
myChart.SeriesCollection(1).XValues = "'Input'!$E$8:$E$19"
myChart.SeriesCollection(1).Values = "'Input'!$C$8:$C$19"
'reset the chart size and position
myChart.PlotArea.Height = 500
myChart.PlotArea.Width = 800
myChart.PlotArea.Top = 10
myChart.PlotArea.Left = 80
'x axis scale
myChart.HasAxis(xlValue) = True
myChart.Axes(xlCategory).Select
myChart.Axes(xlCategory).MinimumScale = 0
myChart.Axes(xlCategory).MaximumScale = 250
myChart.Axes(xlCategory).MajorUnit = 50
'chart name
myChart.HasAxis(xlValue) = True
ChartName = Split(myChart.Name, " ")
ChartName = ChartName(1) & " " & ChartName(2)
'y axis scale
ActiveSheet.ChartObjects(ChartName).Activate
myChart.Axes(xlValue).Select
myChart.Axes(xlValue).MinimumScale = 0
myChart.Axes(xlValue).MaximumScale = 100
'y axis crosses x axis at
myChart.HasAxis(xlValue) = True
ActiveSheet.ChartObjects(ChartName).Activate
myChart.Axes(xlCategory).Select
myChart.Axes(xlCategory).CrossesAt = 125
'remove legend
myChart.HasAxis(xlValue) = True
ActiveSheet.ChartObjects(ChartName).Activate
myChart.Axes(xlValue).Select
myChart.SetElement (msoElementLegendNone)
'add data label of bubbles
myChart.SetElement (msoElementDataLabelCenter)
'add x axis title
myChart.SetElement (msoElementPrimaryCategoryAxisTitleAdjacentToAxis)
myChart.Axes(xlCategory, xlPrimary).AxisTitle.Select
myChart.Axes(xlCategory, xlPrimary).AxisTitle.Text = "Horizontal Title"
'add y axis title
myChart.SetElement (msoElementPrimaryValueAxisTitleRotated)

```

```

myChart.Axes(xlValue, xlPrimary).AxisTitle.Select
myChart.Axes(xlValue, xlPrimary).AxisTitle.Text = "Vertical Title"
'select plot area
myChart.PlotArea.Select
'select all data label
myChart.SeriesCollection(1).DataLabels.Select
'select all bubbles
myChart.SeriesCollection(1).Select
'size is width or area
myChart.ChartGroups(1).SizeRepresents = xlSizelsWidth
myChart.ChartGroups(1).SizeRepresents = xlSizelsArea
'select all data label and position below
myChart.SeriesCollection(1).DataLabels.Select
Selection.Position = xlLabelPositionBelow
'select data label of bubble 9
myChart.SeriesCollection(1).Points(9).DataLabel.Select
'add title above chart
myChart.SetElement (msoElementChartTitleAboveChart)
'chart style of bubbles
myChart.ChartStyle = 31
End Function

```

Private Sub test()

```

Sheets("Input").Activate
Dim Rng As Range
Sheets("Input").Activate
Set Rng = Sheets("Input").Range(Cells(8, 3), Cells(10, 5))

Dim myChart As Excel.Chart
'Set mychart = Excel.Application.Charts.Add
Sheets("Chart13").Activate
Set myChart = Excel.Application.Charts(1)
myChart.ChartType = Excel.XlChartType.xlBubble3DEffect

Dim mySeriesCollection As Excel.SeriesCollection
Set mySeriesCollection = myChart.SeriesCollection
Dim mySeries As Excel.Series

myChart.PlotArea.Width = 560
myChart.PlotArea.Height = 400
myChart.PlotArea.Top = 10
myChart.PlotArea.Left = 80

For Each mySeries In mySeriesCollection
    mySeriesCollection.Item(1).Delete
Next

'mySeriesCollection.Add Source:=Rng
Sheets("Input").Activate
Charts("Chart13").SeriesCollection.Extend _
Source:=Sheets("Input").Range("B8:B19")

```

```
Charts("Chart13").SeriesCollection.Add _  
Source:=ActiveWorkbook.Worksheets("Input").Range("B8:D19")
```

```
'MySeries.Border.LineStyle = Excel.XlLineStyle.xlContinuous  
Sheets("Chart13").myChart.SeriesCollection.Extend Rng
```

```
mySeries.XValues = "=Sheet1!R2C1:R201C1"  
mySeries.Values = "=Sheet1!R2C2:R201C2"  
mySeries.Name = """"Name""""
```

```
myChart.Legend.Top = 340  
myChart.Legend.Left = 490  
myChart.Location Where:=Excel.XlChartLocation.xlLocationAsNewSheet, Name:=""  
Excel.Application.ActiveWindow.Zoom = 100  
'Excel.Application.Name = "Name"
```

```
' MySeries.Trendlines.Add Type:=Excel.XlTrendlineType.xlPolynomial, Order:=jieshu  
' , Forward:=0, Backward:=0, DisplayEquation:=True, DisplayRSquared:= _  
' True  
mySeries.Trendlines(1).DataLabel.Top = 20  
mySeries.Trendlines(1).DataLabel.Left = 250
```

```
End Sub  
Private Sub sdjkllds()  
Dim Mycount As String  
Dim myChtObj As ChartObject  
Dim srsNew As SeriesCollection  
Dim myChart As Chart  
Dim shp As Shape  
Range("A:A").Select  
Mycount = ActiveSheet.UsedRange.Rows.Count
```

```
'THIS DELETES ALL CHARTS FROM THE ACTIVE WORKSHEET  
'With Worksheets(1)  
With ActiveSheet  
For Each shp In .Shapes  
  
If shp.Type = msoChart Then shp.Delete  
  
Next shp  
End With
```

'THIS IS FOR CHART 1 ACTUAL SVL

Set myChtObj = ActiveSheet.ChartObjects.Add(Left:=1300, Width:=800, Top:=0, Height:=425)

myChtObj.Name = "mychart"

' myChtObj.Chart.ChartType = xlLineMarkers

'SVL Forecast

With myChtObj

.Chart.SetSourceData Source:=Range("q2:" & "q" & Mycount)

.Chart.SeriesCollection(1).XValues = Range("d2:" & "e" & Mycount)

'ActiveSheet.ChartObjects(1).Activate

'ActiveChart.Axes(xlCategory).Select

With .Chart.Axes(xlCategory).TickLabels

.Alignment = xlCenter

.Offset = 100

.ReadingOrder = xlContext

.Orientation = 90

End With

.Chart.SeriesCollection(1).Name = "SVL Forecast"

.Chart.SeriesCollection(1).ChartType = xlColumnClustered

.Chart.ChartTitle.Text = "Comparing SVL Forecast VsActual"

With .Chart.ChartTitle.Characters.Font

.Name = "Arial"

.FontStyle = "Bold"

.Size = 19

End With

End With

'SVL Actual

With myChtObj

.Chart.SeriesCollection.Add \_

Source:=Range("p2:" & "p" & Mycount)

.Chart.SeriesCollection(2).Name = "SVL Actual"

```
.Chart.SeriesCollection(1).ChartType = xlLineMarkers  
End With
```

```
End Sub
```

```
Private Sub ChangeSeries1Color()  
    Dim myChartObject As ChartObject  
    Dim myChart As Chart  
    Dim mySeries As Series  
    Dim MyChartFormat As ChartFormat  
    'Dim MyFillFormat As Line  
    'Dim MyColorFormat As ColorFormat  
    ' Create the objects  
    Set myChartObject = ActiveSheet.ChartObjects(1)  
    Set myChart = myChartObject.Chart  
    Set mySeries = myChart.SeriesCollection(1)  
    Set MyChartFormat = mySeries.Format  
    'Set MyFillFormat = MyChartFormat.Line  
    'Set MyColorFormat = MyFillFormat.ForeColor  
    ' Change the color  
    MyChartFormat.Line.ForeColor.RGB = vbGreen  
End Sub
```

```
Sub charts2()  
'So erstelle ich derzeit die einzelnen Charts im VBA:  
Charts.Add  
ActiveChart.ChartType = xlColumnClustered  
ActiveChart.SetSourceData Source:=Rng, PlotBy:=xlColumns  
ActiveChart.SeriesCollection(1).XValues = "=" & ChartSheet & "!R3C" & colnum + 1 & ":R" & Bins + 2  
& "C" & colnum + 1  
ActiveChart.SeriesCollection(1).Name = "Probability"  
ActiveChart.Location Where:=xlLocationAsObject, Name:=ChartSheet  
ActiveChart.HasTitle = False  
ActiveChart.Axes(xlCategory, xlPrimary).HasTitle = False  
ActiveChart.Axes(xlValue, xlPrimary).HasTitle = True  
ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "Probability"  
ActiveChart.Axes(xlValue, xlPrimary).TickLabels.NumberFormat = "0.000"  
ActiveChart.HasLegend = False  
With ActiveChart  
    Set srsNew = .SeriesCollection.NewSeries  
  
    With srsNew  
        .Name = "Frequency"  
        .Values = "=" & ChartSheet & "!R407C" & colnum + 1 & ":R" & Bins + 406 & "C" & colnum + 1  
        If ChartSheet = "ChartsAssum" Then  
            .Interior.Color = assumcol  
        Else  
            .Interior.Color = forecol  
        End If  
        .Interior.Pattern = xlSolid
```

```
.AxisGroup = 2  
End With
```

```
With .ChartGroups(2)  
.Overlap = 100  
.GapWidth = 0  
.HasSeriesLines = False  
.VaryByCategories = False  
End With
```

```
With .ChartArea  
.Interior.ColorIndex = xlNone  
.Border.Weight = xlThin  
.Border.LineStyle = xlNone  
End With
```

```
With .Axes(xlCategory)  
.CrossesAt = 1  
.TickLabelSpacing = Int(Bins / 25) + 1  
.TickMarkSpacing = 1  
.AxisBetweenCategories = True  
.ReversePlotOrder = False  
    With .TickLabels  
.Alignment = xlCenter  
.Offset = 0  
.ReadingOrder = xlContext  
.Orientation = xlUpward  
If ((arrcharts(11, instance) > 1000) Or (arrcharts(10, instance) < -1000)) Then  
.NumberFormat = "# ##0"  
End If  
End With  
End With
```

```
With .Axes(xlValue, xlSecondary)  
.HasTitle = True  
.AxisTitle.Characters.Text = "Frequency"  
.Crosses = xlAutomatic  
.ReversePlotOrder = False  
.ScaleType = xlLinear  
.DisplayUnit = xlNone  
End With
```

```
.PlotArea.Interior.ColorIndex = 2
```

```
Set srsNew = .SeriesCollection.NewSeries
```

```
With srsNew  
.Name = ""Left Wing""  
.Values = "" & ChartSheet & "!R306C" & colnum + 1 & ":R" & Bins + 305 & "C" & colnum + 1  
.AxisGroup = 2  
.Shadow = False  
.InvertIfNegative = False
```



```

With .Interior
If ChartSheet = "ChartsAssum" Then
.Color = assumleftcol
Else
.Color = foreleftcol
End If
.Pattern = xlSolid
End With

```

```

With .Border
.Weight = xlThin
.LineStyle = xlAutomatic
End With

```

End With

```
Set srsNew = .SeriesCollection.NewSeries
```

```

With srsNew
.Name = """"Right Wing""""
.Values = "=" & ChartSheet & "!R508C" & colnum + 1 & ":R" & Bins + 507 & "C" & colnum + 1
.AxisGroup = 2
.Shadow = False
.InvertIfNegative = False

```

```

With .Interior
If ChartSheet = "ChartsAssum" Then
.Color = assumrightcol
Else
.Color = forerightcol
End If
.Pattern = xlSolid
End With

```

```

With .Border
.Weight = xlThin
.LineStyle = xlAutomatic
End With

```

End With

```

.Axes(xlValue, xlPrimary).MinorUnit = .Axes(xlValue, xlSecondary).MinorUnit / TrialRun
'.Axes(xlValue, xlPrimary).MajorUnit = .Axes(xlValue, xlSecondary).MajorUnit / TrialRun
.Axes(xlValue, xlSecondary).MajorUnitIsAuto = False
.Axes(xlValue, xlSecondary).MinorUnitIsAuto = False
.Axes(xlValue, xlPrimary).MinimumScaleIsAuto = False
.Axes(xlValue, xlPrimary).MaximumScaleIsAuto = False
.Axes(xlValue, xlSecondary).MinimumScaleIsAuto = False
.Axes(xlValue, xlPrimary).MinimumScale = 0
.Axes(xlValue, xlSecondary).MinimumScale = 0
ValueMax = .Axes(xlValue, xlSecondary).MaximumScale
.Axes(xlValue, xlPrimary).MaximumScale = ValueMax / TrialRun
End With

```

'wobei das Problem mit dem Stapelspeicher erst auftritt wenn ich in jedes der Charts noch folgende Serie hinzufüge (davon gibt es 2).

```
With Workbooks(chbk).Worksheets(chsh).ChartObjects("Chart " & charnum).Chart  
Set srsNew = .SeriesCollection.NewSeries
```

```
With srsNew
```

```
.Name = """"Mean""""  
.ChartType = xlXYScatter  
.MarkerBackgroundColorIndex = xlAutomatic  
.MarkerForegroundColorIndex = xlAutomatic  
.MarkerStyle = xlNone  
.Smooth = False  
.MarkerSize = 5  
.Shadow = False  
.ErrorBar Direction:=xlY, Include:=xlMinusValues, Type:=xlPercent, Amount:=100  
.ApplyDataLabels AutoText:=True, LegendKey:= _  
False, ShowSeriesName:=True, ShowCategoryName:=False, ShowValue:=False, _  
ShowPercentage:=False, ShowBubbleSize:=False, Separator:="" & Chr(10) & ""
```

```
With .ErrorBars.Border
```

```
.LineStyle = xlContinuous  
.ColorIndex = Color  
.Weight = xlMedium  
End With
```

```
.ErrorBars.EndStyle = xlNoCap
```

```
.XValues = MeanPos
```

```
.Values = Workbooks(chbk).Worksheets(chsh).ChartObjects("Chart " &  
charnum).Chart.Axes(xlValue, xlPrimary).MaximumScale
```

```
With .DataLabels
```

```
.AutoScaleFont = True
```

```
With .Font
```

```
.Name = "Arial"  
.FontStyle = "Fett"  
.Size = 8.5  
.Strikethrough = False  
.Superscript = False  
.Subscript = False  
.OutlineFont = False  
.Shadow = False  
.Underline = xlUnderlineStyleNone  
.ColorIndex = Color  
.Background = xlAutomatic  
End With
```

```
With .Interior
```

```
.ColorIndex = 2  
.PatternColorIndex = 2  
.Pattern = xlSolid  
End With
```

```

        With .Border
            .Weight = xlThin
            .LineStyle = xlSolid
        End With

        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
        .ReadingOrder = xlContext
        .Position = LabelPos
        .Orientation = xlHorizontal

    End With
End With
End With

End Sub

'With ActiveSheet.ChartObjects(1).Chart
' Do
'   .SeriesCollection.NewSeries
'   .SeriesCollection(I).Name = sh.Range(sh.Cells(Z, 1), sh.Cells(Z, 1))
'   .SeriesCollection(I).XValues = sh.Range(sh.Cells(Z, 1), sh.Cells(Z, 1))
'   .SeriesCollection(I).Values = sh.Range(sh.Cells(Z, 2), sh.Cells(Z, 2))
'   .SeriesCollection(I).BubbleSizes = "=Tabelle1!R" & Z & "C3" & ":R" & Z & "C3"
'   .SeriesCollection(I).ChartType = xlBubble3DEffect
'   Z = Z + 1
'   I = I + 1
' Loop Until sh.Cells(Z, 1) = ""
'End With

'Sub BubbleChart()
'   'define variables for workbook, chart, and chart series
'   MyBook = ActiveWorkbook.Name
'   myChart = ActiveChart.Name
'   MySeries = ActiveChart.SeriesCollection(1).Formula
'
'   'define variables for worksheet and chart series reference
'   StartVal = InStr(InStr(1, MySeries, "(") + 1, MySeries, ",") + 1
'   EndSheetVal = InStr(StartVal, MySeries, "!")
'   mysheet = Mid(MySeries, StartVal, EndSheetVal - StartVal)
'   EndVal = InStr(StartVal, MySeries, ",")
'   mysource = Mid(MySeries, StartVal, EndVal - StartVal)
'
'   If InStr(mysheet, "'") Then           'strip out apostrophe
'       mysheet = Mid(mysheet, 2, Len(mysheet) - 2) 'if sheet name has a
'   End If                               'space
'
'   'begin loop to add data labels to chart

```

```

' Counter = 1
' For Each xlItem In Range(mysource)
'     xLabel = xlItem.Offset(0, -1).Value
'     ActiveChart.SeriesCollection(1).Points(Counter).HasDataLabel _
'         = True
'     ActiveChart.SeriesCollection(1).Points(Counter).DataLabel.Text _
'         = xLabel
'     Counter = Counter + 1
' Next xlItem
'
' 'create oval on worksheet (used for chart bubbles)
' Workbooks(MyBook).Sheets(mysheet).Activate
' ActiveSheet.Ovals.Add(335.25, 12.75, 52.5, 52.5).Select
' Application.ScreenUpdating = False
' MyOval = ActiveSheet.DrawingObjects.Name
'
' 'get values from worksheet to compute bubble sizes
' Set MyBubbleRange = Range(mysource).Offset(0, 3)
'
' 'begin loop to compute bubble size and add to chart data point
' For Counter = 1 To MyBubbleRange.Count
'     BubbleValue = MyBubbleRange(Counter) * 50
'     ActiveSheet.DrawingObjects(MyOval).Select
'     With Selection
'         .Width = BubbleValue
'         .Height = BubbleValue
'     End With
'     Selection.Copy
'     Workbooks(MyBook).Sheets(myChart).Activate
'     ActiveChart.SeriesCollection(1).Points(Counter).Select
'     Selection.Paste
'
'     'select worksheet
'     MyBubbleRange.Parent.Activate
' Next Counter
'
' 'activate chartsheet
' ActiveWorkbook.Sheets(myChart).Activate
'
' 'remove oval from worksheet
' ActiveWorkbook.Sheets(mysheet).DrawingObjects(MyOval).Delete
' End Sub

```

'1. In a new worksheet in Microsoft Excel, enter the following values:

```

' A1:    B1: Gross Revenues C1: Net Income D1: # of Plants
' A2: East B2: 831191      C2: 35427   D2: 26
' A3: West B3: 622199      C3: 54263   D3: 13
' A4: North B4: 153794     C4: 80881   D4: 40
' A5: South B5: 711327     C5: 33872   D5: 35

```

' 2. Select the range B1:C5. From the Insert menu, choose Chart, As New Sheet. In the chart wizard choose the following:

- Step 2: Choose XY(Scatter)
  - Step 3: choose Type 1
  - Step 4: choose Data Series in Columns; Use first 1 column(s) for X data; Use first 1 row(s) for legend text.
  - Step 5: choose No for Add a legend; enter titles is optional.
3. Activate the worksheet and type the following formula in cell E2:  
`=D2/MAX($D$2:$D$5)` 4. Select cells E2:E5 and click Fill Down on the Edit menu. This formula calculates the number of plants for each region relative to the total number of plants in all four regions.
5. Select the chart sheet and run the macro. Click Macro on the Tools menu, click BubbleChart, and click Run.

'Option Explicit

,

'Public Sub Blasendiagramm()

'Dim s%, i%, sh As Worksheet

'Set sh = Sheets("erg")

's = 1: i = 1

,

'Application.ScreenUpdating = False

'Charts.Add

'ActiveChart.Location Where:=xlLocationAsObject, Name:="erg"

,

'With ActiveSheet.ChartObjects(1).Chart

' Do

' .SeriesCollection.NewSeries

' .SeriesCollection(i).Name = sh.Range(sh.Cells(3, s), sh.Cells(3, s))

' .SeriesCollection(i).XValues = sh.Range(sh.Cells(18, s), sh.Cells(20, s))

' .SeriesCollection(i).Values = sh.Range(sh.Cells(4, 2), sh.Cells(6, 2))

' .SeriesCollection(i).BubbleSizes = "=erg!R4C" & s & ":R6C" & s

' If i > 1 Then .SeriesCollection(i).ChartType = xlBubble3DEffect

' s = s + 1

' i = i + 1

' Loop Until sh.Cells(3, s) = ""

,

' .ChartArea.Interior.ColorIndex = 15

' .PlotArea.Interior.ColorIndex = 2

' .SeriesCollection(5).Interior.ColorIndex = 43

' .SeriesCollection(9).Interior.ColorIndex = 33

,

' With .Legend.LegendEntries(1).LegendKey

' .Delete

' .Delete

' End With

' .Axes(xlCategory).MinimumScale = 0

' .Axes(xlValue).MinimumScale = 0

'End With

,

'ActiveSheet.[b4:l15].Copy ActiveSheet.[b4]

'With Application

' .SendKeys "{esc}"

' .ScreenUpdating = True

'End With

```

'
'End Sub

'Function z_GetChartName(Sh_chart As String, Optional ChartTitle As String)
'  Sheets(Sh_chart).Activate
'  Cells(1, 1).Activate
'  Dim myChart As Excel.Chart
'  Dim N As Integer: N = 1
'  Dim nofCharts As Integer
'  nofCharts = Sheets(Sh_chart).ChartObjects.Count
'  If nofCharts = 1 Then
'    Sheets(Sh_chart).ChartObjects(N).Activate
'    Set myChart = ActiveChart
'    ChartName = Split(myChart.Name, " ")
'    ChartName = ChartName(1) & " " & ChartName(2)
'    ChartObjectName = Sheets(Sh_chart).ChartObjects(N).Name
'  Else
'    For N = 1 To nofCharts
'      Sheets(Sh_chart).ChartObjects(N).Visible = True
'      Sheets(Sh_chart).ChartObjects(N).Activate
'      If ActiveChart.HasTitle = True And ChartTitle <> Empty Then
'        If ActiveChart.ChartTitle.Text = ChartTitle Then
'          Set myChart = ActiveChart
'          ChartName = Split(myChart.Name, " ")
'          ChartName = ChartName(1) & " " & ChartName(2)
'          ChartObjectName = Sheets(Sh_chart).ChartObjects(N).Name
'        End If
'      ElseIf ActiveChart.HasTitle = False And ChartTitle <> Empty Then
'        ActiveChart.HasTitle = True
'        If ActiveChart.ChartTitle.Text = ChartTitle Then
'          Set myChart = ActiveChart
'          ChartName = Split(myChart.Name, " ")
'          ChartName = ChartName(1) & " " & ChartName(2)
'          ChartObjectName = Sheets(Sh_chart).ChartObjects(N).Name
'        End If
'        ActiveChart.HasTitle = False
'      ElseIf ActiveChart.HasTitle = True And ChartTitle = Empty Then
'        Set myChart = ActiveChart
'        Dim myChartObject As ChartObject
'        Set myChartObject = myChart.Parent
'        ChartName = Split(myChart.Name, " ")
'        ChartName = ChartName(1) & " " & ChartName(2)
'        ChartObjectName = myChartObject.Name
'      ElseIf ActiveChart.HasTitle = False And ChartTitle = Empty Then
'
'      Else
'
'    End If
'  Next
'  For Each Chart_ In ActiveSheet.ChartObjects
'    Chart_.Activate
'    If ActiveChart.HasTitle = True And ChartTitle <> Empty Then
'      If ActiveChart.ChartTitle.Text = ChartTitle Then

```

```

"        Set myChart = ActiveChart
"    End If
"    ElseIf ActiveChart.HasTitle = False And ChartTitle <> Empty Then
"        ActiveChart.HasTitle = True
"        If ActiveChart.ChartTitle.Text = ChartTitle Then
"            Set myChart = ActiveChart
"        End If
"        ActiveChart.HasTitle = False
"    Else
"    End If
" Next
' End If

```

```
Private Sub m_GetChartValues()
```

```
End Sub
```

```
Private Function z_GetChartValues()
```

```

'This macro will retrieve the source data from a chart in excel
'This works for charts where the source data has been lost or
'damaged.
'Simply select the chart and run the macro - make sure to create a
'separate worksheet titled "ChartData" first though.
'

```

```

    Dim NumberOfRows As Integer
    Dim X As Object
    Counter = 2
    NumberOfRows = UBound(ActiveChart.SeriesCollection(1).Values)
    Worksheets("ChartData").Cells(1, 1) = "X Values"
    With Worksheets("ChartData")
        .Range(.Cells(2, 1), _
        .Cells(NumberOfRows + 1, 1)) = _
        Application.Transpose(ActiveChart.SeriesCollection(1).XValues)
    End With
    For Each X In ActiveChart.SeriesCollection
        Worksheets("ChartData").Cells(1, Counter) = X.Name
        With Worksheets("ChartData")
            .Range(.Cells(2, Counter), _
            .Cells(NumberOfRows + 1, Counter)) = _
            Application.Transpose(X.Values)
        End With
        Counter = Counter + 1
    Next
End Function

```

```

Private Sub m_Chart_DeleteFormatting()
    Worksheets("Sheet1").ChartObjects(1).Chart.ChartArea.ClearFormats
End Sub

```

```

Private Sub m_Chart_AddFormats()
    Dim LabelRange As Range

```

```

    Sheets("Input").Select
    Set LabelRange = Sheets("Input").Range(Cells(11, 2), Cells(30, 2))
    Call z_Chart_AddFormats("Input", "Chart", LabelRange)
End Sub

```

```

Private Sub z_Chart_AddFormats(Sh_Sorce As String, Sh_Chart As String, LabelRange As Range)
    Sheets(Sh_Chart).Activate
    Cells(1, 1).Activate
    Dim myChartObject As Excel.ChartObject
    Dim myChart As Excel.Chart
    Dim mySeriesCollection As Excel.SeriesCollection
    Dim mySeries As Excel.Series
    Dim myPoints As Excel.Points
    Dim myPoint As Excel.Point
    Dim Iter_Chart As Integer
    Dim iter_Series As Integer
    Dim iter_Point As Integer
    'iterate through all chartObjects(and its chart and seriesCollection) in Sh_chart
    Dim nofCharts As Integer
    nofCharts = Sheets(Sh_Chart).ChartObjects.Count
    For Iter_Chart = 1 To nofCharts
        Set myChartObject = Sheets(Sh_Chart).ChartObjects(Iter_Chart)
        Set myChart = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart
        Set mySeriesCollection = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection
        'iterate through all series and change the properties of all points (bubbles) of each series
        Dim nofSeries As Integer
        nofSeries = myChart.SeriesCollection.Count
        For iter_Series = 1 To nofSeries
            Set mySeries = Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series)
            mySeries.Interior.Color = RGB(75, 50, 50)
            mySeries.Axes(xlCategory).MinimumScale = 0
            mySeries.Axes(xlValue).MinimumScale = 0
            'iterate through all points (bubbles) of a series and change the properties of the points
            Set myPoints =
            Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series).Points
            nofPoints = myPoints.Count
            For iter_Point = 1 To nofPoints
                Set myPoint =
                Sheets(Sh_Chart).ChartObjects(Iter_Chart).Chart.SeriesCollection(iter_Series).Points(iter_Point)
                On Error Resume Next
                myPoints(iter_Point).ClearFormats
                'myPoints(iter_Point).MarkerBackgroundColor
                'myPoints(iter_Point).MarkerForegroundColor
                On Error GoTo 0
            Next
        Next
    Next
End Sub

```

```

Private Sub m_ProtectChartObject()
    'Protect chart object size and position
    Sheets(Sh_Chart).ChartObjects(N).ProtectChartObject = False
    a = Sheets(Sh_Chart).ChartObjects(N).ProtectChartObject

```



```
'dont move or size with cell

'lock (takes only effect if sheet is protected)
Sheets(Sh_Chart).ChartObjects(N).Protect
End Sub
```

## **File: Fit Supply to Demand 2012**

```
'To do:
*****

'A. Make the changes in the source tables (or pivots) so that for each ratio there is only 1 line in the
pivots:
'-----
'1. corn: Barley and Wheat and probably others; CostDriver-Demand inconsistent; change DEMAND
source (or pivot) to Barley and Wheat
'2. facility: CostDriver-Demand; GH passive and GH active; change BOTH sources (or pivots) to GH
'3. facility: CostDriver-Demand inconsistent; for nurseries - inbreds change in BOTH SOURCES the
facility to "not used"
'4. facility: Hughes-Demand inconsistent; for yield trial and nurseries-observation change in DEMAND
SOURCE the facility to "not used"
'5. all attributes: fill the (Blanks) with values in the DEMAND SOURCE
'Before starting:
'-----
'1. year: set filter in Hughes pivot
'2. all attributes: remove all auto filters in the pivots
'3. set the flags to true or false in Main_Fitting under "2. Invoke Fitting Algorithms (VBA Outputs)"
'4. set the parameters accordingly for layout changes in Main_Fitting under "1. Define the layout
parameters"
*****
```

Sub Main\_Fitting()

```
'1. Define the layout parameters
*****

'1.1 Layout Demand (FT input for Hughes input)
'-----
Dim Sh_DemandPivot_ForHughesInput As Worksheet
Set Sh_DemandPivot_ForHughesInput = Sheets("demand pivot ex FP temp - H")
Dim Col_DemandPivot_ForHughesInput_A_Country As Long
Col_DemandPivot_ForHughesInput_A_Country = z_GetColumnIndex("country", 4,
Sh_DemandPivot_ForHughesInput.Name)
Dim Col_DemandPivot_ForHughesInput_C_Crop As Long
Col_DemandPivot_ForHughesInput_C_Crop = z_GetColumnIndex("crop", 4,
Sh_DemandPivot_ForHughesInput.Name)
Dim Col_DemandPivot_ForHughesInput_D_Activity As Long
Col_DemandPivot_ForHughesInput_D_Activity = z_GetColumnIndex("activity / trial type", 4,
Sh_DemandPivot_ForHughesInput.Name)
Dim Col_DemandPivot_ForHughesInput_E_Facility As Long
Col_DemandPivot_ForHughesInput_E_Facility = z_GetColumnIndex("facility adjusted", 4,
Sh_DemandPivot_ForHughesInput.Name)
```

```

Dim Col_DemandPivot_ForHughesInput_F_SumOfTotalNoPlots As Long
    Col_DemandPivot_ForHughesInput_F_SumOfTotalNoPlots = z_GetColumnIndex("Sum of total
no. plots (or no. detailed facility ex column S)", 4, Sh_DemandPivot_ForHughesInput.Name)
Dim Col_DemandPivot_ForHughesInput_G_SumOfTotalNoPlants As Long
    Col_DemandPivot_ForHughesInput_G_SumOfTotalNoPlants = z_GetColumnIndex("Sum of total
no. plants (calculated)", 4, Sh_DemandPivot_ForHughesInput.Name)
Dim Row_DemandPivot_ForHughesInput_From As Long
    Row_DemandPivot_ForHughesInput_From = 5
Dim Row_DemandPivot_ForHughesInput_To As Long
    Row_DemandPivot_ForHughesInput_To = z_RowSize(1,
Sh_DemandPivot_ForHughesInput.Name)
'Layout DemandPivot compression
'-----

```

```

Dim Layout_DemandPivot_ForHughesInput(1 To 8) As Variant
    Layout_DemandPivot_ForHughesInput(1) = Col_DemandPivot_ForHughesInput_A_Country
    Layout_DemandPivot_ForHughesInput(2) = Col_DemandPivot_ForHughesInput_C_Crop
    Layout_DemandPivot_ForHughesInput(3) = Col_DemandPivot_ForHughesInput_D_Activity
    Layout_DemandPivot_ForHughesInput(4) = Col_DemandPivot_ForHughesInput_E_Facility
    Layout_DemandPivot_ForHughesInput(5) =
Col_DemandPivot_ForHughesInput_F_SumOfTotalNoPlots
    Layout_DemandPivot_ForHughesInput(6) =
Col_DemandPivot_ForHughesInput_G_SumOfTotalNoPlants
    Layout_DemandPivot_ForHughesInput(7) = Row_DemandPivot_ForHughesInput_From
    Layout_DemandPivot_ForHughesInput(8) = Row_DemandPivot_ForHughesInput_To

```

'1.2 Layout Demand (FT input for CostDriver input)

```

'-----
Dim Sh_DemandPivot_ForCostDriverInput As Worksheet
Set Sh_DemandPivot_ForCostDriverInput = Sheets("demand pivot ex FP temp - CD")
Dim Col_DemandPivot_ForCostDriverInput_A_Country As Long
    Col_DemandPivot_ForCostDriverInput_A_Country = z_GetColumnIndex("country", 4,
Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_C_Crop As Long
    Col_DemandPivot_ForCostDriverInput_C_Crop = z_GetColumnIndex("crop adjusted", 4,
Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_D_Activity As Long
    Col_DemandPivot_ForCostDriverInput_D_Activity = z_GetColumnIndex("activity / trial type", 4,
Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_E_Facility As Long
    Col_DemandPivot_ForCostDriverInput_E_Facility = z_GetColumnIndex("facility adjusted", 4,
Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_F_SumOfTotalNoPlots As Long
    Col_DemandPivot_ForCostDriverInput_F_SumOfTotalNoPlots = z_GetColumnIndex("Sum of total
no. plots (or no. detailed facility ex column S)", 4, Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_G_SumOfTotalNoPlants As Long
    Col_DemandPivot_ForCostDriverInput_G_SumOfTotalNoPlants = z_GetColumnIndex("Sum of
total no. plants (calculated)", 4, Sh_DemandPivot_ForCostDriverInput.Name)
Dim Row_DemandPivot_ForCostDriverInput_From As Long
    Row_DemandPivot_ForCostDriverInput_From = 5
Dim Row_DemandPivot_ForCostDriverInput_To As Long
    Row_DemandPivot_ForCostDriverInput_To = z_RowSize(1,
Sh_DemandPivot_ForCostDriverInput.Name)
'Layout DemandPivot compression

```

```

'-----
Dim Layout_DemandPivot_ForCostDriverInput(1 To 8) As Variant
    Layout_DemandPivot_ForCostDriverInput(1) =
Col_DemandPivot_ForCostDriverInput_A_Country
    Layout_DemandPivot_ForCostDriverInput(2) = Col_DemandPivot_ForCostDriverInput_C_Crop
    Layout_DemandPivot_ForCostDriverInput(3) = Col_DemandPivot_ForCostDriverInput_D_Activity
    Layout_DemandPivot_ForCostDriverInput(4) = Col_DemandPivot_ForCostDriverInput_E_Facility
    Layout_DemandPivot_ForCostDriverInput(5) =
Col_DemandPivot_ForCostDriverInput_F_SumOfTotalNoPlots
    Layout_DemandPivot_ForCostDriverInput(6) =
Col_DemandPivot_ForCostDriverInput_G_SumOfTotalNoPlants
    Layout_DemandPivot_ForCostDriverInput(7) = Row_DemandPivot_ForCostDriverInput_From
    Layout_DemandPivot_ForCostDriverInput(8) = Row_DemandPivot_ForCostDriverInput_To

```

### '1.3 Layout Supply (Hughes input)

```

'-----
Dim Sh_SupplyHughesPivot As Worksheet
Set Sh_SupplyHughesPivot = Sheets("Hughes' pivot")
Dim Col_SupplyHughesPivot_A_Country As Long
    Col_SupplyHughesPivot_A_Country = z_GetColumnIndex("Country of trial achievement", 3,
Sh_SupplyHughesPivot.Name)
Dim Col_SupplyHughesPivot_C_Crop As Long
    Col_SupplyHughesPivot_C_Crop = z_GetColumnIndex("Crop type2 (detailed)", 3,
Sh_SupplyHughesPivot.Name)
Dim Col_SupplyHughesPivot_D_Activity As Long
    Col_SupplyHughesPivot_D_Activity = z_GetColumnIndex("Type of Field activity", 3,
Sh_SupplyHughesPivot.Name)
Dim Col_SupplyHughesPivot_E_SumOfTotalNoPlots As Long
    Col_SupplyHughesPivot_E_SumOfTotalNoPlots = z_GetColumnIndex("Sum of Nb Real plots", 3,
Sh_SupplyHughesPivot.Name)
Dim Row_SupplyHughesPivot_From As Long
    Row_SupplyHughesPivot_From = 5
Dim Row_SupplyHughesPivot_To As Long
    Row_SupplyHughesPivot_To = z_RowSize(1, Sh_SupplyHughesPivot.Name)
'Layout SupplyHughesPivot compression
'-----

```

```

Dim Layout_SupplyHughesPivot(1 To 6) As Variant
    Layout_SupplyHughesPivot(1) = Col_SupplyHughesPivot_A_Country
    Layout_SupplyHughesPivot(2) = Col_SupplyHughesPivot_C_Crop
    Layout_SupplyHughesPivot(3) = Col_SupplyHughesPivot_D_Activity
    Layout_SupplyHughesPivot(4) = Col_SupplyHughesPivot_E_SumOfTotalNoPlots
    Layout_SupplyHughesPivot(5) = Row_SupplyHughesPivot_From
    Layout_SupplyHughesPivot(6) = Row_SupplyHughesPivot_To

```

### '1.4 Layout Supply (CostDriver input)

```

'-----
Dim Sh_SupplyCostDriverPivot As Worksheet
Set Sh_SupplyCostDriverPivot = Sheets("Cost Driver pivot")
Dim Col_SupplyCostDriverPivot_A_Country As Long
    Col_SupplyCostDriverPivot_A_Country = z_GetColumnIndex("country", 3,
Sh_SupplyCostDriverPivot.Name)
Dim Col_SupplyCostDriverPivot_C_Crop As Long

```

```

    Col_SupplyCostDriverPivot_C_Crop = z_GetColumnIndex("crop adjusted", 3,
Sh_SupplyCostDriverPivot.Name)
    Dim Col_SupplyCostDriverPivot_D_Activity As Long
    Col_SupplyCostDriverPivot_D_Activity = z_GetColumnIndex("cost driver", 3,
Sh_SupplyCostDriverPivot.Name)
    Dim Col_SupplyCostDriverPivot_E_Facility As Long
    Col_SupplyCostDriverPivot_E_Facility = z_GetColumnIndex("facility adjusted", 3,
Sh_SupplyCostDriverPivot.Name)
    Dim Col_SupplyCostDriverPivot_F_Units As Long
    Col_SupplyCostDriverPivot_F_Units = z_GetColumnIndex("unit for all regions", 3,
Sh_SupplyCostDriverPivot.Name)
    Dim Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits As Long
    Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits = z_GetColumnIndex("Sum of no. of units
supplied", 3, Sh_SupplyCostDriverPivot.Name)
    Dim Row_SupplyCostDriverPivot_From As Long
    Row_SupplyCostDriverPivot_From = 5
    Dim Row_SupplyCostDriverPivot_To As Long
    Row_SupplyCostDriverPivot_To = z_RowSize(1, Sh_SupplyCostDriverPivot.Name)
'Layout SupplyCostDriverPivot compression
'-----
Dim Layout_SupplyCostDriverPivot(1 To 8) As Variant
    Layout_SupplyCostDriverPivot(1) = Col_SupplyCostDriverPivot_A_Country
    Layout_SupplyCostDriverPivot(2) = Col_SupplyCostDriverPivot_C_Crop
    Layout_SupplyCostDriverPivot(3) = Col_SupplyCostDriverPivot_D_Activity
    Layout_SupplyCostDriverPivot(4) = Col_SupplyCostDriverPivot_E_Facility
    Layout_SupplyCostDriverPivot(5) = Col_SupplyCostDriverPivot_F_Units
    Layout_SupplyCostDriverPivot(6) = Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits
    Layout_SupplyCostDriverPivot(7) = Row_SupplyCostDriverPivot_From
    Layout_SupplyCostDriverPivot(8) = Row_SupplyCostDriverPivot_To

'1.4 Layout Fitting (VBA Output)
'-----
Dim Col_Fitting_B_Country As Long
    Col_Fitting_B_Country = 2
Dim Row_Fitting_6_Crop As Long
    Row_Fitting_6_Crop = 6
Dim Row_Fitting_From As Long
    Row_Fitting_From = 8
Dim Row_Fitting_To As Long
    Row_Fitting_To = 55
Dim Col_Fitting_From As Long
    Col_Fitting_From = 3
Dim Col_Fitting_To As Long
    Col_Fitting_To = 41
'Layout Fitting compression
'-----
Dim Layout_Fitting(1 To 6) As Variant
    Layout_Fitting(1) = Col_Fitting_B_Country
    Layout_Fitting(2) = Row_Fitting_6_Crop
    Layout_Fitting(3) = Row_Fitting_From
    Layout_Fitting(4) = Row_Fitting_To
    Layout_Fitting(5) = Col_Fitting_From
    Layout_Fitting(6) = Col_Fitting_To

```

## '2. Invoke Fitting Algorithms (VBA Outputs)

\*\*\*\*\*

Dim Flag\_YieldTrials As Boolean

Flag\_YieldTrials = True

Dim Flag\_NurseriesObservation As Boolean

Flag\_NurseriesObservation = True

Dim Flag\_NurseriesCrossesField As Boolean

Flag\_NurseriesCrossesField = True

Dim Flag\_NurseriesCrossesGreenHouses As Boolean

Flag\_NurseriesCrossesGreenHouses = True

Dim Flag\_NurseriesInbreds As Boolean

Flag\_NurseriesInbreds = True

'Demand (FT input)

'Supply (Hughes input)

'-----

' 2.1.1 fitting yield trials

'-----

' Dim Sh\_Fitting\_YieldTrials As Worksheet

' Set Sh\_Fitting\_YieldTrials = Sheets("fitting yield trials")

' '!Layout\_Fitting(0) = Sh\_Fitting\_YieldTrials

' 'activity search string

'-----

' Dim Value\_DemandPivot\_YieldTrials\_D\_Activity As String

' Value\_DemandPivot\_YieldTrials\_D\_Activity = "yield trial"

' Dim Value\_SupplyHughesPivot\_YieldTrials\_D\_Activity As String

' Value\_SupplyHughesPivot\_YieldTrials\_D\_Activity = "yield trial"

' 'facility search string

'-----

' Dim Value\_DemandPivot\_YieldTrials\_E\_Facility As Variant

' Value\_DemandPivot\_YieldTrials\_E\_Facility = "not used"

' Dim Value\_SupplyHughesPivot\_YieldTrials\_E\_Facility As Variant

' Value\_SupplyHughesPivot\_YieldTrials\_E\_Facility = Empty

' 'invoke fitting algorithm

'-----

' Dim Unit As String

' Unit = "plot"

' Dim Supply As String

' Supply = "Hughes Pivot"

' If Flag\_YieldTrials Then

' Call z\_Fitting\_Clear(Sh\_Fitting\_YieldTrials, Layout\_Fitting)

' Call z\_Fitting(Sh\_Fitting\_YieldTrials, \_

' Sh\_DemandPivot\_ForHughesInput, \_

' Sh\_SupplyHughesPivot, \_

' Sh\_SupplyCostDriverPivot, \_

' Layout\_Fitting, \_

' Layout\_DemandPivot\_ForHughesInput, \_

' Layout\_SupplyHughesPivot, \_

' Layout\_SupplyCostDriverPivot, \_

' Unit, \_

' Supply, \_

' Value\_DemandPivot\_YieldTrials\_D\_Activity, \_

```

'           Value_SupplyHughesPivot_YieldTrials_D_Activity, _
'           Value_DemandPivot_YieldTrials_E_Facility, _
'           Value_SupplyHughesPivot_YieldTrials_E_Facility)
' End If

' 2.1.2 fitting nurseries-observation
' -----
' Dim Sh_Fitting_NurseriesObservation As Worksheet
' Set Sh_Fitting_NurseriesObservation = Sheets("fitting nurseries-observation")
' !Layout_Fitting(0) = Sh_Fitting_NurseriesObservation
'   'activity search string
'   '-----
'   Dim Value_DemandPivot_NurseriesObservation_D_Activity As String
'   Value_DemandPivot_NurseriesObservation_D_Activity = "nurseries - observation"
'   Dim Value_SupplyHughesPivot_NurseriesObservation_D_Activity As String
'   Value_SupplyHughesPivot_NurseriesObservation_D_Activity = "observation"
'   'facility search string
'   '-----
'   Dim Value_DemandPivot_NurseriesObservation_E_Facility As Variant
'   Value_DemandPivot_NurseriesObservation_E_Facility = "not used"
'   Dim Value_SupplyHughesPivot_NurseriesObservation_E_Facility As Variant
'   Value_SupplyHughesPivot_NurseriesObservation_E_Facility = Empty
'   'invoke fitting algorithm
'   '-----
'   Unit = "plot"
'   Supply = "Hughes Pivot"
' If Flag_NurseriesObservation Then
'   Call z_Fitting_Clear(Sh_Fitting_NurseriesObservation, Layout_Fitting)
'   Call z_Fitting(Sh_Fitting_NurseriesObservation, _
'     Sh_DemandPivot_ForHughesInput, _
'     Sh_SupplyHughesPivot, _
'     Sh_SupplyCostDriverPivot, _
'     Layout_Fitting, _
'     Layout_DemandPivot_ForHughesInput, _
'     Layout_SupplyHughesPivot, _
'     Layout_SupplyCostDriverPivot, _
'     Unit, _
'     Supply, _
'     Value_DemandPivot_NurseriesObservation_D_Activity, _
'     Value_SupplyHughesPivot_NurseriesObservation_D_Activity, _
'     Value_DemandPivot_NurseriesObservation_E_Facility, _
'     Value_SupplyHughesPivot_NurseriesObservation_E_Facility)
' End If

'Demand (FT input)
'Supply (CostDriver input)
'-----
' 2.2.1 fitting nurseries-crosses field
' -----
' Dim Sh_Fitting_NurseriesCrossesField As Worksheet
' Set Sh_Fitting_NurseriesCrossesField = Sheets("fitting nurseries-crosses field")
' !Layout_Fitting(0) = Sh_Fitting_NurseriesCrossesField
'   'activity search string

```

```

'-----
Dim Value_DemandPivot_NurseriesCrossesField_D_Activity As String
    Value_DemandPivot_NurseriesCrossesField_D_Activity = "nurseries - crosses"
Dim Value_SupplyCostDriverPivot_NurseriesCrossesField_D_Activity As String
    Value_SupplyCostDriverPivot_NurseriesCrossesField_D_Activity = "nurseries - crosses"
'facility search string
'-----

Dim Value_DemandPivot_NurseriesCrossesField_E_Facility As Variant
    Value_DemandPivot_NurseriesCrossesField_E_Facility = "open field"
Dim Value_SupplyCostDriverPivot_NurseriesCrossesField_E_Facility As Variant
    Value_SupplyCostDriverPivot_NurseriesCrossesField_E_Facility = "open field"
'invoke fitting algorithm
'-----

    Dim Unit As String
    Dim Supply As String
    Unit = "row"
    Supply = "Cost Driver Pivot"
If Flag_NurseriesCrossesField Then
    Call z_Fitting_Clear(Sh_Fitting_NurseriesCrossesField, Layout_Fitting)
    Call z_Fitting(Sh_Fitting_NurseriesCrossesField, _
        Sh_DemandPivot_ForCostDriverInput, _
        Sh_SupplyHughesPivot, _
        Sh_SupplyCostDriverPivot, _
        Layout_Fitting, _
        Layout_DemandPivot_ForCostDriverInput, _
        Layout_SupplyHughesPivot, _
        Layout_SupplyCostDriverPivot, _
        Unit, _
        Supply, _
        Value_DemandPivot_NurseriesCrossesField_D_Activity, _
        Value_SupplyCostDriverPivot_NurseriesCrossesField_D_Activity, _
        Value_DemandPivot_NurseriesCrossesField_E_Facility, _
        Value_SupplyCostDriverPivot_NurseriesCrossesField_E_Facility)
End If

'2.2.2 fitting nurseries-crosses GH
'-----
Dim Sh_Fitting_NurseriesCrossesGreenHouses As Worksheet
Set Sh_Fitting_NurseriesCrossesGreenHouses = Sheets("fitting nurseries-crosses GH")
'!Layout_Fitting(0) = Sh_Fitting_NurseriesCrossesGreenHouses
'activity search string
'-----

Dim Value_DemandPivot_NurseriesCrossesGreenHouses_D_Activity As String
    Value_DemandPivot_NurseriesCrossesGreenHouses_D_Activity = "nurseries - crosses"
Dim Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_D_Activity As String
    Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_D_Activity = "nurseries -
crosses"
'facility search string
'-----

Dim Value_DemandPivot_NurseriesCrossesGreenHouses_E_Facility As Variant
    Value_DemandPivot_NurseriesCrossesGreenHouses_E_Facility = "GH" ' "GH Passive" and "GH
active"
Dim Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_E_Facility As Variant

```

```

        Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_E_Facility = "GH" ' "GH Passive"
and "GH active"

```

```

'invoke fitting algorithm

```

```

'-----

```

```

    Unit = "cross"

```

```

    Supply = "Cost Driver Pivot"

```

```

If Flag_NurseriesCrossesGreenHouses Then

```

```

    Call z_Fitting_Clear(Sh_Fitting_NurseriesCrossesGreenHouses, Layout_Fitting)

```

```

    Call z_Fitting(Sh_Fitting_NurseriesCrossesGreenHouses, _

```

```

        Sh_DemandPivot_ForCostDriverInput, _

```

```

        Sh_SupplyHughesPivot, _

```

```

        Sh_SupplyCostDriverPivot, _

```

```

        Layout_Fitting, _

```

```

        Layout_DemandPivot_ForCostDriverInput, _

```

```

        Layout_SupplyHughesPivot, _

```

```

        Layout_SupplyCostDriverPivot, _

```

```

        Unit, _

```

```

        Supply, _

```

```

        Value_DemandPivot_NurseriesCrossesGreenHouses_D_Activity, _

```

```

        Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_D_Activity, _

```

```

        Value_DemandPivot_NurseriesCrossesGreenHouses_E_Facility, _

```

```

        Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_E_Facility)

```

```

End If

```

```

'2.2.3.fitting nurseries-inbreds

```

```

'-----

```

```

Dim Sh_Fitting_NurseriesInbreds As Worksheet

```

```

Set Sh_Fitting_NurseriesInbreds = Sheets("fitting nurseries-inbreds")

```

```

'!Layout_Fitting(0) = Sh_Fitting_NurseriesInbreds

```

```

'activity search string

```

```

'-----

```

```

Dim Value_DemandPivot_NurseriesInbreds_D_Activity As String

```

```

    Value_DemandPivot_NurseriesInbreds_D_Activity = "nurseries - inbreds"

```

```

Dim Value_SupplyCostDriverPivot_NurseriesInbreds_D_Activity As String

```

```

    Value_SupplyCostDriverPivot_NurseriesInbreds_D_Activity = "nurseries - inbreds"

```

```

'facility search string

```

```

'-----

```

```

Dim Value_DemandPivot_NurseriesInbreds_E_Facility As Variant

```

```

    Value_DemandPivot_NurseriesInbreds_E_Facility = "not used"

```

```

Dim Value_SupplyCostDriverPivot_NurseriesInbreds_E_Facility As Variant

```

```

    Value_SupplyCostDriverPivot_NurseriesInbreds_E_Facility = "not used"

```

```

'invoke fitting algorithm

```

```

'-----

```

```

    Unit = "plant"

```

```

    Supply = "Cost Driver Pivot"

```

```

If Flag_NurseriesInbreds Then

```

```

    Call z_Fitting_Clear(Sh_Fitting_NurseriesInbreds, Layout_Fitting)

```

```

    Call z_Fitting(Sh_Fitting_NurseriesInbreds, _

```

```

        Sh_DemandPivot_ForCostDriverInput, _

```

```

        Sh_SupplyHughesPivot, _

```

```

        Sh_SupplyCostDriverPivot, _

```

```

        Layout_Fitting, _

```

```

        Layout_DemandPivot_ForCostDriverInput, _

```



```

        Layout_SupplyHughesPivot, _
        Layout_SupplyCostDriverPivot, _
        Unit, _
        Supply, _
        Value_DemandPivot_NurseriesInbreds_D_Activity, _
        Value_SupplyCostDriverPivot_NurseriesInbreds_D_Activity, _
        Value_DemandPivot_NurseriesInbreds_E_Facility, _
        Value_SupplyCostDriverPivot_NurseriesInbreds_E_Facility)
End If

```

#### '2.2.4 fitting nurseries-observation

```

'-----
Dim Sh_Fitting_NurseriesObservation As Worksheet
Set Sh_Fitting_NurseriesObservation = Sheets("fitting nurseries-observation")
'!Layout_Fitting(0) = Sh_Fitting_NurseriesObservation
'activity search string
'-----
Dim Value_DemandPivot_NurseriesObservation_D_Activity As String
    Value_DemandPivot_NurseriesObservation_D_Activity = "nurseries - observation"
Dim Value_SupplyCostDriverPivot_NurseriesObservation_D_Activity As String
    Value_SupplyCostDriverPivot_NurseriesObservation_D_Activity = "nurseries - observation"
'facility search string
'-----
Dim Value_DemandPivot_NurseriesObservation_E_Facility As Variant
    Value_DemandPivot_NurseriesObservation_E_Facility = "not used"
Dim Value_SupplyCostDriverPivot_NurseriesObservation_E_Facility As Variant
    Value_SupplyCostDriverPivot_NurseriesObservation_E_Facility = "not used"
'invoke fitting algorithm
'-----
    Unit = "plot"
    Supply = "Cost Driver Pivot"
If Flag_NurseriesObservation Then
    Call z_Fitting_Clear(Sh_Fitting_NurseriesObservation, Layout_Fitting)
    Call z_Fitting(Sh_Fitting_NurseriesObservation, _
        Sh_DemandPivot_ForCostDriverInput, _
        Sh_SupplyHughesPivot, _
        Sh_SupplyCostDriverPivot, _
        Layout_Fitting, _
        Layout_DemandPivot_ForCostDriverInput, _
        Layout_SupplyHughesPivot, _
        Layout_SupplyCostDriverPivot, _
        Unit, _
        Supply, _
        Value_DemandPivot_NurseriesObservation_D_Activity, _
        Value_SupplyCostDriverPivot_NurseriesObservation_D_Activity, _
        Value_DemandPivot_NurseriesObservation_E_Facility, _
        Value_SupplyCostDriverPivot_NurseriesObservation_E_Facility)
End If

```

#### '2.2.5 fitting yield trials

```

'-----
Dim Sh_Fitting_YieldTrials As Worksheet

```

```

Set Sh_Fitting_YieldTrials = Sheets("fitting yield trials")
'!Layout_Fitting(0) = Sh_Fitting_YieldTrials
'activity search string
'-----
Dim Value_DemandPivot_YieldTrials_D_Activity As String
Value_DemandPivot_YieldTrials_D_Activity = "yield trial"
Dim Value_SupplyHughesPivot_YieldTrials_D_Activity As String
Value_SupplyHughesPivot_YieldTrials_D_Activity = "yield trial"
'facility search string
'-----
Dim Value_DemandPivot_YieldTrials_E_Facility As Variant
Value_DemandPivot_YieldTrials_E_Facility = "not used"
Dim Value_SupplyHughesPivot_YieldTrials_E_Facility As Variant
Value_SupplyHughesPivot_YieldTrials_E_Facility = "not used"
'invoke fitting algorithm
'-----
Unit = "plot"
Supply = "Cost Driver Pivot"
If Flag_YieldTrials Then
Call z_Fitting_Clear(Sh_Fitting_YieldTrials, Layout_Fitting)
Call z_Fitting(Sh_Fitting_YieldTrials, _
Sh_DemandPivot_ForCostDriverInput, _
Sh_SupplyHughesPivot, _
Sh_SupplyCostDriverPivot, _
Layout_Fitting, _
Layout_DemandPivot_ForCostDriverInput, _
Layout_SupplyHughesPivot, _
Layout_SupplyCostDriverPivot, _
Unit, _
Supply, _
Value_DemandPivot_YieldTrials_D_Activity, _
Value_SupplyHughesPivot_YieldTrials_D_Activity, _
Value_DemandPivot_YieldTrials_E_Facility, _
Value_SupplyHughesPivot_YieldTrials_E_Facility)
End If
End Sub

Public Function z_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, _
Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Output datatype change from Variant
Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer
'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
'Sheets(Sh).Activate
Dim Sht As Worksheet
Set Sht = Sheets(Sh)
'find column name
Dim Cl As Range

```

```

'Set Cl = Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole)
Set Cl = Sht.Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole)
If Cl Is Nothing Then GoTo NameExpectedNotExistent
'find column index
'CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexStr = Cl.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
'Output
z_GetColumnIndex = ColIndex
Exit Function
OptionalArgument:
Resume Next
NameExpectedNotExistent:
ColIndex = 0
Stop 'only in test mode
z_GetColumnIndex = ColIndex
End Function

Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet
'Sheets(Sh).Activate
'Determine the row size
z_RowSize = If(IsEmpty(Sheets(Sh).Cells(1048576, SearchCol)), Sheets(Sh).Cells(1048576, SearchCol).End(xlUp).Row, 1048576)
End Function

Function z_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Row, Output: column with the last entry in that row
'SearchCol datatype changed from integer
'Activate the Sheet
'Sheets(Sh).Activate
'Determine the col size
z_ColSize = If(IsEmpty(Sheets(Sh).Cells(SearchRow, 16384)), Sheets(Sh).Cells(SearchRow, 16384).End(xlToLeft).Column, 16384)
End Function

Sub m_TrimCells_FP()
    Call z_TrimCells(ActiveSheet.Name, Range(Cells(1, 1), Cells(5000, 40)))
End Sub
Sub m_TrimCells_CD()
    Call z_TrimCells(ActiveSheet.Name, Range(Cells(1, 1), Cells(500, 10)))
End Sub

Private Function z_TrimCells(Sh As String, Optional ByRef Rng As Range)

```

```

'Sheets("Sheet1").Activate
'Call z_TrimCells("Sheet1", Range(Cells(3, 1), Cells(3, 5)))

Sheets(Sh).Activate
If Rng Is Nothing Then
    'select all
    Cells.Select
    Set Rng = Selection.Cells
Else
    'take the input range
End If

Dim MyRngIndices() As Long
MyRngIndices = z_RangeToIndices(Rng)

On Error Resume Next
For Row = MyRngIndices(0) To MyRngIndices(2)
    For Col = MyRngIndices(1) To MyRngIndices(3)
        With Excel.WorksheetFunction
            Cells(Row, Col) = .Trim(.Clean(Cells(Row, Col)))
        End With
    Next Col
Next Row
On Error GoTo 0

End Function
*****Get Range Indices
Private Function z_RangeToIndices(ByRef Rng As Range) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim RangeIndices(0 To 3) As Long
    Dim CellsArray() As String
    Dim sAddr As String
    sAddr = Rng.Address(ReferenceStyle:=xlR1C1)
    CellsArray = Split(sAddr, ":")
    Dim CellIndicesUL() As Long
    On Error GoTo RangelsColumnOrRow
    CellIndicesUL = z_sCellToIndex(CellsArray(0))
    On Error GoTo 0
    Dim CellIndicesLR() As Long
    On Error GoTo RangelsCell
    CellIndicesLR = z_sCellToIndex(CellsArray(1))
    On Error GoTo 0
    RangeIndices(0) = CellIndicesUL(0)
    RangeIndices(1) = CellIndicesUL(1)
    RangeIndices(2) = CellIndicesLR(0)
    RangeIndices(3) = CellIndicesLR(1)
    z_RangeToIndices = RangeIndices
Exit Function
RangelsCell:
CellIndicesLR = z_sCellToIndex(CellsArray(0))
Resume Next
RangelsColumnOrRow:
Dim RorC As String

```

```

RorC = Left(sAddr, 1)
OneOrMore = InStr(1, sAddr, ":", vbTextCompare)
'only one row or column
If OneOrMore = 0 Then
    If RorC = "C" Then
        RangeIndices(0) = 1
        RangeIndices(1) = z_sColumnToIndex(CellsArray(0))
        RangeIndices(2) = 1048534
        RangeIndices(3) = RangeIndices(0)
    ElseIf RorC = "R" Then
        RangeIndices(0) = z_sRowToIndex(CellsArray(0))
        RangeIndices(1) = 1
        RangeIndices(2) = RangeIndices(0)
        RangeIndices(3) = 16383
    Else
        Stop
    End If
'more than one row or column
Else
    If RorC = "C" Then
        RangeIndices(0) = 1
        RangeIndices(1) = z_sColumnToIndex(CellsArray(0))
        RangeIndices(2) = 1048534
        RangeIndices(3) = z_sColumnToIndex(CellsArray(1))
    ElseIf RorC = "R" Then
        RangeIndices(0) = z_sRowToIndex(CellsArray(0))
        RangeIndices(1) = 1
        RangeIndices(2) = z_sRowToIndex(CellsArray(1))
        RangeIndices(3) = 16383
    Else
        Stop
    End If
End If
z_RangeToIndices = RangeIndices
End Function

```

```

Private Function z_sCellToIndex(ByRef CellIndexStr As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim CellIndexArr() As String 'Splited R1C1 Format
    Dim ColIndex As Integer
    Dim RowIndex As Integer
    Dim CellIndices(0 To 1) As Long
    'find column index
    CellIndexArr = Split(CellIndexStr, "C")
    ColIndex = CInt(CellIndexArr(1))
    CellIndexArr = Split(CellIndexArr(0), "R")
    RowIndex = CInt(CellIndexArr(1))
    CellIndices(0) = RowIndex
    CellIndices(1) = ColIndex
    'Output
    z_sCellToIndex = CellIndices
End Function

```

```

Private Function z_sColumnToIndex(ByRef ColIndexStrLeft As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim ColArray() As String
    ColArray = Split(ColIndexStrLeft, "C")
    z_sColumnToIndex = ColArray(1)
End Function

```

```

Private Function z_sRowToIndex(ByRefRowIndexStrUp As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim RowArray() As String
    RowArray = Split(RowIndexStrUp, "R")
    z_sRowToIndex = RowArray(1)
End Function
*****Get Range Indices

```

```

Function z_Fitting(Sh_Fitting As Worksheet, _
    Sh_DemandPivot As Worksheet, _
    Sh_SupplyHughesPivot As Worksheet, _
    Sh_SupplyCostDriverPivot As Worksheet, _
    Layout_Fitting As Variant, _
    Layout_DemandPivot As Variant, _
    Layout_SupplyHughesPivot As Variant, _
    Layout_SupplyCostDriverPivot As Variant, _
    Unit As String, _
    Supply As String, _
    Value_DemandPivot_D_Activity As String, _
    Value_SupplyPivot_D_Activity As String, _
    Value_DemandPivot_E_Facility As Variant, _
    Value_SupplyPivot_E_Facility As Variant)

'Layout_Fitting decompression
'-----
'!Dim Sh_Fitting As Worksheet
'! Sh_Fitting = Layout_Fitting(0)
Dim Col_Fitting_B_Country As Long
    Col_Fitting_B_Country = Layout_Fitting(1)
Dim Row_Fitting_6_Crop As Long
    Row_Fitting_6_Crop = Layout_Fitting(2)
Dim Row_Fitting_From As Long
    Row_Fitting_From = Layout_Fitting(3)
Dim Row_Fitting_To As Long
    Row_Fitting_To = Layout_Fitting(4)
Dim Col_Fitting_From As Long
    Col_Fitting_From = Layout_Fitting(5)
Dim Col_Fitting_To As Long
    Col_Fitting_To = Layout_Fitting(6)

'Iterate through all rows of the Fitting Matrix
'-----
Dim Row_Fitting_iter As Long
For Row_Fitting_iter = Row_Fitting_From To Row_Fitting_To Step 1

    'Iterate through all columns of the Fitting Matrix

```

```

'-----
Dim Col_Fitting_iter As Long
For Col_Fitting_iter = Col_Fitting_From To Col_Fitting_To Step 1

    'Search Sting (Matrix Dimensions of the Fitting Table)
    Dim SearchString_Fitting(0 To 1) As Variant
    SearchString_Fitting(0) = LCase(Sh_Fitting.Cells(Row_Fitting_iter, Col_Fitting_B_Country))
    SearchString_Fitting(1) = LCase(Sh_Fitting.Cells(Row_Fitting_6_Crop, Col_Fitting_iter))
    'for debug purposes
    If LCase(SearchString_Fitting(0)) = "france" And LCase(SearchString_Fitting(1)) = "sunflower"
Then
        'Stop 'for debug purposes
        End If

    'search in Demand (FT input)
    '-----
    Dim Cell_DemandPivot_SumOfTotalNo As Range
    Set Cell_DemandPivot_SumOfTotalNo = z_SearchIn_DemandPivot( _
        Sh_DemandPivot, _
        Layout_DemandPivot, _
        SearchString_Fitting, _
        Unit, _
        Value_DemandPivot_D_Activity, _
        Value_DemandPivot_E_Facility)

    'search in Supply (Hughes input)
    '-----
    If Supply = "Hughes Pivot" Then
        Dim Cell_SupplyPivot_SumOfTotalNo As Range
        Set Cell_SupplyPivot_SumOfTotalNo = z_SearchIn_Supply_HughesPivot( _
            Sh_SupplyHughesPivot, _
            Layout_SupplyHughesPivot, _
            SearchString_Fitting, _
            Unit, _
            Value_SupplyPivot_D_Activity, _
            Value_SupplyPivot_E_Facility)

        'search in Supply (CostDriver input)
        '-----
        Elself Supply = "Cost Driver Pivot" Then
            Set Cell_SupplyPivot_SumOfTotalNo = z_SearchIn_Supply_CostDriverPivot( _
                Sh_SupplyCostDriverPivot, _
                Layout_SupplyCostDriverPivot, _
                SearchString_Fitting, _
                Unit, _
                Value_SupplyPivot_D_Activity, _
                Value_SupplyPivot_E_Facility)

        Else
            Stop
        End If

    'calculate the ratio demand/supply and write it into the fitting sheet

```

```

'white=2 no demand ,red=3 demand but no supply
'-----

Call z_Calculation(Sh_Fitting, Row_Fitting_iter, Col_Fitting_iter, _
    Cell_DemandPivot_SumOfTotalNo, Cell_SupplyPivot_SumOfTotalNo)

'Set the objects to nothing
'-----

Set Cell_DemandPivot_SumOfTotalNo = Nothing
Set Cell_SupplyPivot_SumOfTotalNo = Nothing

Next Col_Fitting_iter
Next Row_Fitting_iter
End Function

Function z_SearchIn_DemandPivot(Sh_DemandPivot As Worksheet, _
    Layout_DemandPivot As Variant, _
    SearchString_Fitting As Variant, _
    Unit As String, _
    Value_DemandPivot_D_Activity As String, _
    Value_DemandPivot_E_Facility As Variant) As Range

'Layout_Demand decompression
'-----
'!Dim Sh_DemandPivot As Worksheet
'!Set Sh_DemandPivot = Layout_DemandPivot(0)
Dim Col_DemandPivot_A_Country As Long
    Col_DemandPivot_A_Country = Layout_DemandPivot(1)
Dim Col_DemandPivot_C_Crop As Long
    Col_DemandPivot_C_Crop = Layout_DemandPivot(2)
Dim Col_DemandPivot_D_Activity As Long
    Col_DemandPivot_D_Activity = Layout_DemandPivot(3)
Dim Col_DemandPivot_E_Facility As Long
    Col_DemandPivot_E_Facility = Layout_DemandPivot(4)
Dim Col_DemandPivot_F_SumOfTotalNoUnits As Long
    Col_DemandPivot_F_SumOfTotalNoUnits = Layout_DemandPivot(5)
Dim Col_DemandPivot_G_SumOfTotalNoPlants As Long
    Col_DemandPivot_G_SumOfTotalNoPlants = Layout_DemandPivot(6)
Dim Row_DemandPivot_From As Long
    Row_DemandPivot_From = Layout_DemandPivot(7)
Dim Row_DemandPivot_To As Long
    Row_DemandPivot_To = Layout_DemandPivot(8)

'Iterate through all rows starting at the bottom of the pivot table
'-----

Dim Row_DemandPivot
For Row_DemandPivot = Row_DemandPivot_To To Row_DemandPivot_From Step -1

'Check the last column of the pivot table
'-----

Dim Cell_DemandPivot_E_Facility As Range
Set Cell_DemandPivot_E_Facility = Sh_DemandPivot.Cells(Row_DemandPivot,
Col_DemandPivot_E_Facility)
If LCase(Cell_DemandPivot_E_Facility.Value) = LCase(Value_DemandPivot_E_Facility) Then

```



```

'Check the second before last column of the pivot table
'-----
Dim Cell_DemandPivot_D_Activity As Range
Set Cell_DemandPivot_D_Activity = Sh_DemandPivot.Cells(Row_DemandPivot,
Col_DemandPivot_D_Activity)
'have not found the row therefore go one row up
If LCase(Cell_DemandPivot_D_Activity.Value) = Empty Then
    Do Until LCase(Cell_DemandPivot_D_Activity.Value) <> Empty
        Set Cell_DemandPivot_D_Activity = Cell_DemandPivot_D_Activity.Offset(-1, 0)
    Loop
End If
'have found the row therefore step in to check the next column
If LCase(Cell_DemandPivot_D_Activity.Value) = LCase(Value_DemandPivot_D_Activity) Then

'Check the third before last column of the pivot table
'-----
Dim Cell_DemandPivot_C_Crop As Range
Set Cell_DemandPivot_C_Crop = Sh_DemandPivot.Cells(Row_DemandPivot,
Col_DemandPivot_C_Crop)
'have not found the row therefore go one row up
If LCase(Cell_DemandPivot_C_Crop.Value) = Empty Then
    Do Until LCase(Cell_DemandPivot_C_Crop.Value) <> Empty
        Set Cell_DemandPivot_C_Crop = Cell_DemandPivot_C_Crop.Offset(-1, 0)
    Loop
End If
'have found the row therefore step in to check the next column
If LCase(Cell_DemandPivot_C_Crop.Value) = LCase(SearchString_Fitting(1)) Then

'Check the fourth before last column of the pivot table
'-----
Dim Cell_DemandPivot_A_Country As Range
Set Cell_DemandPivot_A_Country = Sh_DemandPivot.Cells(Row_DemandPivot,
Col_DemandPivot_A_Country)
'have not found the row therefore go one row up
If LCase(Cell_DemandPivot_A_Country.Value) = Empty Then
    Do Until LCase(Cell_DemandPivot_A_Country.Value) <> Empty
        Set Cell_DemandPivot_A_Country = Cell_DemandPivot_A_Country.Offset(-1, 0)
    Loop
End If
'have found the row therefore step in to get the needed value
If LCase(Cell_DemandPivot_A_Country.Value) = LCase(SearchString_Fitting(0)) Then

'Output and exit
'-----
If LCase(Unit) = "plot" Or LCase(Unit) = "row" Or LCase(Unit) = "cross" Then
    Dim Cell_DemandPivot_F_SumOfTotalNoUnits As Range
    Set Cell_DemandPivot_F_SumOfTotalNoUnits =
Sh_DemandPivot.Cells(Row_DemandPivot, Col_DemandPivot_F_SumOfTotalNoUnits)
    Set z_SearchIn_DemandPivot = Cell_DemandPivot_F_SumOfTotalNoUnits
ElseIf LCase(Unit) = "plant" Then
    Dim Cell_DemandPivot_G_SumOfTotalNoPlants As Range

```

```

        Set Cell_DemandPivot_G_SumOfTotalNoPlants =
Sh_DemandPivot.Cells(Row_DemandPivot, Col_DemandPivot_G_SumOfTotalNoPlants)
        Set z_SearchIn_DemandPivot = Cell_DemandPivot_G_SumOfTotalNoPlants
    Else
        Stop
    End If
Exit For

End If
End If
End If
End If
Next Row_DemandPivot
End Function

Function z_SearchIn_Supply_HughesPivot(Sh_SupplyHughesPivot As Worksheet, _
    Layout_SupplyHughesPivot As Variant, _
    SearchString_Fitting As Variant, _
    Unit As String, _
    Value_SupplyHughesPivot_D_Activity As String, _
    Value_SupplyHughesPivot_E_Facility As Variant) As Range

'Layout_SupplyHughes decompression
'-----
'!Dim Sh_SupplyHughesPivot As Worksheet
'! Sh_SupplyHughesPivot = Layout_SupplyHughesPivot(0)
Dim Col_SupplyHughesPivot_A_Country As Long
    Col_SupplyHughesPivot_A_Country = Layout_SupplyHughesPivot(1)
Dim Col_SupplyHughesPivot_C_Crop As Long
    Col_SupplyHughesPivot_C_Crop = Layout_SupplyHughesPivot(2)
Dim Col_SupplyHughesPivot_D_Activity As Long
    Col_SupplyHughesPivot_D_Activity = Layout_SupplyHughesPivot(3)
Dim Col_SupplyHughesPivot_E_SumOfTotalNoPlots As Long
    Col_SupplyHughesPivot_E_SumOfTotalNoPlots = Layout_SupplyHughesPivot(4)
Dim Row_SupplyHughesPivot_From As Long
    Row_SupplyHughesPivot_From = Layout_SupplyHughesPivot(5)
Dim Row_SupplyHughesPivot_To As Long
    Row_SupplyHughesPivot_To = Layout_SupplyHughesPivot(6)

'Iterate through all rows starting at the bottom of the pivot table
'-----
Dim Row_SupplyHughesPivot As Long
For Row_SupplyHughesPivot = Row_SupplyHughesPivot_To To Row_SupplyHughesPivot_From
Step -1

    'Check the last column of the pivot table
    '-----
    Dim Cell_SupplyHughesPivot_D_Activity As Range
    Set Cell_SupplyHughesPivot_D_Activity = Sh_SupplyHughesPivot.Cells(Row_SupplyHughesPivot,
Col_SupplyHughesPivot_D_Activity)
    If LCase(Cell_SupplyHughesPivot_D_Activity.Value) =
LCase(Value_SupplyHughesPivot_D_Activity) Then

```

```

'Check the second before last column of the pivot table
'-----
Dim Cell_SupplyHughesPivot_C_Crop As Range
Set Cell_SupplyHughesPivot_C_Crop = Sh_SupplyHughesPivot.Cells(Row_SupplyHughesPivot,
Col_SupplyHughesPivot_C_Crop)
'have not found the row therefore go one row up
If LCase(Cell_SupplyHughesPivot_C_Crop.Value) = Empty Then
    Do Until LCase(Cell_SupplyHughesPivot_C_Crop.Value) <> Empty
        Set Cell_SupplyHughesPivot_C_Crop = Cell_SupplyHughesPivot_C_Crop.Offset(-1, 0)
    Loop
End If
'have found the row therefore step in to check the next column
If LCase(Cell_SupplyHughesPivot_C_Crop.Value) = LCase(SearchString_Fitting(1)) Then

    'Check the third before last column of the pivot table
    '-----
    Dim Cell_SupplyHughesPivot_A_Country As Range
    Set Cell_SupplyHughesPivot_A_Country =
Sh_SupplyHughesPivot.Cells(Row_SupplyHughesPivot, Col_SupplyHughesPivot_A_Country)
    'have not found the row therefore go one row up
    If LCase(Cell_SupplyHughesPivot_A_Country.Value) = Empty Then
        Do Until LCase(Cell_SupplyHughesPivot_A_Country.Value) <> Empty
            Set Cell_SupplyHughesPivot_A_Country = Cell_SupplyHughesPivot_A_Country.Offset(-1,
0)
        Loop
    End If
    'have found the row therefore step in to get the needed value
    If LCase(Cell_SupplyHughesPivot_A_Country.Value) = LCase(SearchString_Fitting(0)) Then

        'Output and exit
        '-----
        Dim Cell_SupplyHughesPivot_E_SumOfTotalNoPlots As Range
        Set Cell_SupplyHughesPivot_E_SumOfTotalNoPlots =
Sh_SupplyHughesPivot.Cells(Row_SupplyHughesPivot,
Col_SupplyHughesPivot_E_SumOfTotalNoPlots)
        Set z_SearchIn_Supply_HughesPivot = Cell_SupplyHughesPivot_E_SumOfTotalNoPlots
        Exit For

    End If
End If
End If
Next Row_SupplyHughesPivot
End Function

Function z_SearchIn_Supply_CostDriverPivot(Sh_SupplyCostDriverPivot As Worksheet, _
    Layout_SupplyCostDriverPivot As Variant, _
    SearchString_Fitting As Variant, _
    Unit As String, _
    Value_SupplyCostDriverPivot_D_Activity As String, _
    Value_SupplyCostDriverPivot_E_Facility As Variant) As Range

'Layout_SupplyCostDriver decompression
'-----

```

```

'!Dim Sh_SupplyCostDriverPivot As Worksheet
'! Sh_SupplyCostDriverPivot = Layout_SupplyCostDriverPivot(0)
Dim Col_SupplyCostDriverPivot_A_Country As Long
    Col_SupplyCostDriverPivot_A_Country = Layout_SupplyCostDriverPivot(1)
Dim Col_SupplyCostDriverPivot_C_Crop As Long
    Col_SupplyCostDriverPivot_C_Crop = Layout_SupplyCostDriverPivot(2)
Dim Col_SupplyCostDriverPivot_D_Activity As Long
    Col_SupplyCostDriverPivot_D_Activity = Layout_SupplyCostDriverPivot(3)
Dim Col_SupplyCostDriverPivot_E_Facility As Long
    Col_SupplyCostDriverPivot_E_Facility = Layout_SupplyCostDriverPivot(4)
Dim Col_SupplyCostDriverPivot_F_Units As Long
    Col_SupplyCostDriverPivot_F_Units = Layout_SupplyCostDriverPivot(5)
Dim Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits As Long
    Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits = Layout_SupplyCostDriverPivot(6)
Dim Row_SupplyCostDriverPivot_From As Long
    Row_SupplyCostDriverPivot_From = Layout_SupplyCostDriverPivot(7)
Dim Row_SupplyCostDriverPivot_To As Long
    Row_SupplyCostDriverPivot_To = Layout_SupplyCostDriverPivot(8)

'Iterate through all rows starting at the bottom of the pivot table
'-----
Dim Row_SupplyCostDriverPivot As Long
For Row_SupplyCostDriverPivot = Row_SupplyCostDriverPivot_To To
Row_SupplyCostDriverPivot_From Step -1

    'Check the last column of the pivot table
    '-----
    Dim Cell_SupplyCostDriverPivot_E_Facility As Range
    Set Cell_SupplyCostDriverPivot_E_Facility =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_E_Facility)
    If LCase(Cell_SupplyCostDriverPivot_E_Facility.Value) =
LCase(Value_SupplyCostDriverPivot_E_Facility) Then

        'Check the second before last column of the pivot table
        '-----
        Dim Cell_SupplyCostDriverPivot_D_Activity As Range
        Set Cell_SupplyCostDriverPivot_D_Activity =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_D_Activity)
        'have not found the row therefore go one row up
        If LCase(Cell_SupplyCostDriverPivot_D_Activity.Value) = Empty Then
            Do Until LCase(Cell_SupplyCostDriverPivot_D_Activity.Value) <> Empty
                Set Cell_SupplyCostDriverPivot_D_Activity =
Cell_SupplyCostDriverPivot_D_Activity.Offset(-1, 0)
            Loop
        End If
        'have found the row therefore step in to check the next column
        If LCase(Cell_SupplyCostDriverPivot_D_Activity.Value) =
LCase(Value_SupplyCostDriverPivot_D_Activity) Then

            'Check the third before last column of the pivot table
            '-----
            Dim Cell_SupplyCostDriverPivot_C_Crop As Range

```

```

        Set Cell_SupplyCostDriverPivot_C_Crop =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_C_Crop)
        'have not found the row therefore go one row up
        If LCase(Cell_SupplyCostDriverPivot_C_Crop.Value) = Empty Then
            Do Until LCase(Cell_SupplyCostDriverPivot_C_Crop.Value) <> Empty
                Set Cell_SupplyCostDriverPivot_C_Crop = Cell_SupplyCostDriverPivot_C_Crop.Offset(-1,
0)
            Loop
        End If
        'have found the row therefore step in to check the next column
        If LCase(Cell_SupplyCostDriverPivot_C_Crop.Value) = LCase(SearchString_Fitting(1)) Then

            'Check the fourth before last column of the pivot table
            '-----
            Dim Cell_SupplyCostDriverPivot_A_Country As Range
            Set Cell_SupplyCostDriverPivot_A_Country =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_A_Country)
            'have not found the row therefore go one row up
            If LCase(Cell_SupplyCostDriverPivot_A_Country.Value) = Empty Then
                Do Until LCase(Cell_SupplyCostDriverPivot_A_Country.Value) <> Empty
                    Set Cell_SupplyCostDriverPivot_A_Country =
Cell_SupplyCostDriverPivot_A_Country.Offset(-1, 0)
                Loop
            End If
            'have found the row therefore step in to get the needed value
            If LCase(Cell_SupplyCostDriverPivot_A_Country) = LCase(SearchString_Fitting(0)) Then

                'Output and exit
                '-----
                Dim Cell_SupplyCostDriverPivot_F_Units As Range
                Set Cell_SupplyCostDriverPivot_F_Units =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_F_Units)
                If LCase(Unit) = LCase(Cell_SupplyCostDriverPivot_F_Units.Value) Then
                    Dim Cell_SupplyCostDriverPivot_G_SumOfTotalNoUnits As Range
                    Set Cell_SupplyCostDriverPivot_G_SumOfTotalNoUnits =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot,
Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits)
                    Set z_SearchIn_Supply_CostDriverPivot =
Cell_SupplyCostDriverPivot_G_SumOfTotalNoUnits
                    Exit For
                Else
                    Stop 'error: wrong unit in pivot table
                End If
            End If
        End If
    End If
End If
Next Row_SupplyCostDriverPivot
End Function

Function z_Calculation(Sh_Fitting As Worksheet, Row_Fitting_iter As Long, Col_Fitting_iter As Long, _
Cell_DemandPivot_SumOfTotalNo As Range, Cell_SupplyPivot_SumOfTotalNo As
Range)

```

'calculate the ratio demand/supply and write it into the fitting sheet

'black=1,white=2,red=3,green=4,blue=5,yellow=6,brown=53

-----

'Demand found

If Not Cell\_DemandPivot\_SumOfTotalNo Is Nothing Then

    'Supply found

    If Not Cell\_SupplyPivot\_SumOfTotalNo Is Nothing Then

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter) = \_

        Cell\_DemandPivot\_SumOfTotalNo.Value / Cell\_SupplyPivot\_SumOfTotalNo.Value \* 100

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Interior.ColorIndex = 2

    'Supply not found

    Else

        'Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Value = "S-NA"

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Interior.ColorIndex = 5

    End If

'Demand not found

Else

    'Supply found

    If Not Cell\_SupplyPivot\_SumOfTotalNo Is Nothing Then

        'Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Value = "D-NA"

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Interior.ColorIndex = 53

    'Supply not found

    Else

        'Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Value = "S&D-NA"

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Interior.ColorIndex = 2

    End If

End If

End Function

## **File: FP template 0000 GENERAL VERSION**

Private Sub Workbook\_Open()

    On Error GoTo UnexpectedError:

    Debug.Print "Workbook\_Open\_b " & Now() & " " & LastTime & " " & LastInteraction & " " & LastOnTime & " " & NextOnTime

    'MsgBox ("")

    'Stop

    'initialisation

    AbortEvent\_Workbook\_BeforeClose = False

    count\_CloseAttempts = 0 'if AbortEvent\_Workbook\_BeforeClose is True and the WS still open

    AbortEvent\_Worksheet\_Change = False

    Flag\_WhenOpendNotCheckedOut = False

    Flag\_HideSheets = False ' will be set below after unhiding, copying and hiding the sheets

    AbortMandatoryCells\_InEvent\_Workbook\_BeforeClose = False

    'check out the workbook from the teamspace (set the check out flag)

    'Wbk\_teamspacePathAndName =?

    Wbk\_teamspacePathAndName = ThisWorkbook.FullName

```

'This version has been already checked out manually and is connected
If Workbooks(ThisWorkbook.Name).CanCheckIn = True Then
    'do nothing
'This version is not checked out or not connected
Else
    answer = MsgBox("Would you like to check out the document from teamspace?", vbQuestion +
vbYesNo, "VBA Message")
    'Do not try to check out
    If answer = vbNo Then
        'do nothing
        Flag_WhenOpendNotCheckedOut = True
    'Try to check out
    ElseIf answer = vbYes Then
        Call z_CheckOut
    End If
End If

'unprotect all worksheets and vba
Sheets("FP template").Activate
ActiveSheet.Unprotect Password:="fpsheets"
Sheets("DataSetBackup").Activate
ActiveSheet.Unprotect Password:="fpsheets"
Sheets("Dropdown Lists").Activate
ActiveSheet.Unprotect Password:="fpsheets"

'clear autofilter
Sheets("FP template").Activate
On Error Resume Next
Sheets("FP template").ShowAllData
On Error GoTo UnexpectedError:

'__
Dim ColSize As Long
ColSize = z_ColSize(1, "FP template")
'__

'restore the cell formats
'Stop
'Column 7 and 22 may be prefilled---> not prefilled!!!
Dim NextEmptyRow As Long
Dim NextEmptyRow1 As Long
Dim NextEmptyRow2 As Long
Dim NextEmptyRow3 As Long
'__
NextEmptyRow1 = z_LastWrittenRow("DataSetBackup", 2, ColSize, 10000) + 1
'z_LastWrittenRow("DataSetBackup", 2, 6, 10000) + 1
NextEmptyRow2 = z_LastWrittenRow("DataSetBackup", 2, ColSize, 10000) + 1
'z_LastWrittenRow("DataSetBackup", 8, 21, 10000) + 1
NextEmptyRow3 = z_LastWrittenRow("DataSetBackup", 2, ColSize, 10000) + 1
'z_LastWrittenRow("DataSetBackup", 23, ColSize, 10000) + 1
'__
NextEmptyRow = NextEmptyRow1
If NextEmptyRow1 >= NextEmptyRow2 Then

```

```

If NextEmptyRow1 >= NextEmptyRow3 Then
    NextEmptyRow = NextEmptyRow1
Else
    NextEmptyRow = NextEmptyRow3
End If
Else
    If NextEmptyRow2 >= NextEmptyRow3 Then
        NextEmptyRow = NextEmptyRow2
    Else
        NextEmptyRow = NextEmptyRow3
    End If
End If
'number/text/date
Call z_FormatCellNumbers("FP template", NextEmptyRow)
'colors, borders etc. from DataSetBackup
Sheets("DataSetBackup").Activate
'__
Call z_CopyRange("Formats", "DataSetBackup",
Sheets("DataSetBackup").Range(Sheets("DataSetBackup").Cells(1, 1),
Sheets("DataSetBackup").Cells(10000, ColSize)), _
    "FP template", Sheets("FP template").Cells(1, 1))
'Column width and row high
' Call z_ChangeColWidth(16, "FP template", 1, 14) 'normalbreite
' Call z_ChangeColWidth(28, "FP template", 15, 15) 'Spezialbreite für "activity / trial type"
' Call z_ChangeColWidth(41, "FP template", 16, 16) 'Spezialbreite für "activity / trial type (detailed)"
' Call z_ChangeColWidth(37, "FP template", 17, 17) 'Spezialbreite für "facility"
' Call z_ChangeColWidth(16, "FP template", 18, ColSize) 'normalbreite
Dim Col As Long
For Col = 1 To ColSize
    Dim String_in As String
    Dim String_out As String
    Dim String_arr As Variant
    Dim Width_ As Double
    String_in = Right(Cells(3, Col), 15)
    String_out = z_FindAndReplaceInString(String_in, _
        Array("[", "]"), Array("@", "@"))
    String_arr = Split(String_out, "@")
    If UBound(String_arr) > 1 Then
        Width_ = String_arr(UBound(String_arr) - 1)
        Call z_ChangeColWidth(Width_, "FP template", Col, Col)
    End If
Next
'__
Call z_ChangeRowHeight(60, "FP template", 1, 1)
Call z_ChangeRowHeight(64.5, "FP template", 2, 2)
Call z_ChangeRowHeight(30, "FP template", 3, 3)
Call z_ChangeRowHeight(36, "FP template", 4, 4)
Call z_ChangeRowHeight(15, "FP template", 5, 10000)
'restore the formulas (too slow: improve the function)
'*****
'Call z_Insert_ProductFormula("U", "T", "W", "X", "V", 23, 25, _
    "unique identifier", 1, "FP template")
'*****

```



```

'unprotect the workbook (necessary to unhide sheets)
ThisWorkbook.Unprotect ("fpsheets") 'will be protected again below after unhiding and copying
and hiding sheets

'unhide worksheets
Sheets("FP template").Visible = True
Sheets("Dropdown Lists").Visible = True
Sheets("DataSetBackup").Visible = True 'will be unhidden again below after copying
Sheets("UserNames").Visible = True 'will be unhidden again below after copying
Sheets("DataSetCheckOutVersion").Visible = True 'will be unhidden again below after copying

'flat value copy
Call z_CopySheet("FP template", "DataSetCheckOutVersion")
'copy
Sheets("FP template").Activate
'__
If ColSize > 2 Then
    Call z_CopyRange("Values", "FP template", Sheets("FP template").Range(Cells(6, 2), Cells(10000,
ColSize)), _
        "DataSetBackup", Sheets("DataSetBackup").Cells(6, 2))
End If
'__
'clear
Sheets("UserNames").Activate
Sheets("UserNames").Range(Cells(5, 2), Cells(10000, 16384)).ClearContents

'protect all worksheets and vba
Sheets("FP template").Activate
Cells.Select
Selection.Locked = True
ActiveSheet.Protect Password:="fpsheets", DrawingObjects:=True, Contents:=True,
Scenarios:=True, _
    AllowFormattingCells:=True, AllowFormattingColumns:=True, _
    AllowFormattingRows:=True, AllowSorting:=True, AllowFiltering:=True, _
    AllowUsingPivotTables:=True
Sheets("DataSetBackup").Activate
ActiveSheet.Protect Password:="fpsheets", DrawingObjects:=True, Contents:=True,
Scenarios:=True, _
    AllowFormattingCells:=True, AllowFormattingColumns:=True, _
    AllowFormattingRows:=True, AllowSorting:=True, AllowFiltering:=True, _
    AllowUsingPivotTables:=True
Sheets("Dropdown Lists").Activate
Cells.Select
Selection.Locked = True
ActiveSheet.Protect Password:="fpsheets", DrawingObjects:=True, Contents:=True,
Scenarios:=True, _
    AllowFormattingCells:=True, AllowFormattingColumns:=True, _
    AllowFormattingRows:=True, AllowSorting:=True, AllowFiltering:=True, _
    AllowUsingPivotTables:=True
ActiveSheet.Cells(1, 1).Select

'hide worksheets again

```

```

Sheets("DataSetCheckOutVersion").Visible = False
Sheets("DataSetBackup").Visible = False
Sheets("UserNames").Visible = False

'set the hide flag again
Flag_HideSheets = True

'protect workbook again
ThisWorkbook.Protect Password:="fpsheets", Structure:=True, Windows:=False

'initialisation
GoodBy_Flag = False
Dim InitialTime As Date
InitialTime = Now()
LastTime = InitialTime
LastInteraction = InitialTime
LastOnTime = InitialTime
NextOnTime = InitialTime + AcceptedDuration

'set the first OnTime
Dim Macro As String
Dim NextOnTime_ As Date
Macro = "z_CheckWhetherToSetANewTime"
NextOnTime_ = NextOnTime
'*****

'Call z_SetNewOnTime(NextOnTime_, Macro)
'*****

'select the sheet
Sheets("Dropdown Lists").Activate

'show and position the userform with the intro
UserForm1.Show (vbModeless)

'refresh (remove the copy ranges) and set the cursor
'Stop
ActiveWorkbook.RefreshAll
'Call z_SetCursor("FP template", 2, ColSize, 10000, 2)
Sheets("FP template").Activate
Sheets("FP template").Cells(NextEmptyRow, 2).Select

Debug.Print "Workbook_Open_e " & Now() & " " & LastTime & " " & LastInteraction & " " &
LastOnTime & " " & NextOnTime
Exit Sub
UnexpectedError:
ThisWorkbook.Close True
End Sub

Private Sub Workbook_Activate()
On Error GoTo UnexpectedError:
'someone is working

```

```

    Debug.Print "Workbook_Activate_b " & Now() & " " & LastTime & " " & LastInteraction & " " &
LastOnTime & " " & NextOnTime
    LastInteraction = Now()
    Debug.Print "Workbook_Activate_e " & Now() & " " & LastTime & " " & LastInteraction & " " &
LastOnTime & " " & NextOnTime
Exit Sub
UnexpectedError:
    ThisWorkbook.Close True
End Sub

```

```

Private Sub Workbook_Deactivate()
    On Error GoTo UnexpectedError:
    'someone is working
    Debug.Print "Workbook_Deactivate_b " & Now() & " " & LastTime & " " & LastInteraction & " " &
LastOnTime & " " & NextOnTime
    LastInteraction = Now()
    Debug.Print "Workbook_Deactivate_e " & Now() & " " & LastTime & " " & LastInteraction & " " &
LastOnTime & " " & NextOnTime
Exit Sub
UnexpectedError:
    ThisWorkbook.Close True
End Sub

```

```

Private Sub Workbook_SheetChange(ByVal Sh As Object, ByVal Target As Range)
    On Error GoTo UnexpectedError:
    'someone is working
    Debug.Print "Workbook_SheetChange_b " & Now() & " " & LastTime & " " & LastInteraction & " " &
LastOnTime & " " & NextOnTime
    LastInteraction = Now()
    Debug.Print "Workbook_SheetChange_e " & Now() & " " & LastTime & " " & LastInteraction & " " &
LastOnTime & " " & NextOnTime
Exit Sub
UnexpectedError:
    ThisWorkbook.Close True
End Sub

```

```

Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As Range)
    On Error GoTo UnexpectedError:
    'someone is working
    Debug.Print "Workbook_SheetSelectionChange_b " & Now() & " " & LastTime & " " &
LastInteraction & " " & LastOnTime & " " & NextOnTime
    LastInteraction = Now()
    Debug.Print "Workbook_SheetSelectionChange_e " & Now() & " " & LastTime & " " &
LastInteraction & " " & LastOnTime & " " & NextOnTime
Exit Sub
UnexpectedError:
    ThisWorkbook.Close True
End Sub

```

```

Private Sub Workbook_BeforeClose(Cancel As Boolean)
    On Error Resume Next:
    'Abort Event
    'Stop
    If AbortEvent_Workbook_BeforeClose = True Then
        count_CloseAttempts = count_CloseAttempts + 1
        If count_CloseAttempts >= 3 Then

```

```

        'do not exit sub
Else
    'do exit sub
    Cancel = True
    Exit Sub
End If
End If

If AbortMandatoryCells_InEvent_Workbook_BeforeClose = False Then
    'mandatory cells
    Dim MandatoryCells_OK As Boolean
    MandatoryCells_OK = True
    MandatoryCells_OK = z_InteriorColor_MandatoryData
    If MandatoryCells_OK = False Then
        Dim Response As Integer
        Response = MsgBox( _
            "Your rows entered contain empty mandatory cells." & Chr(13) & Chr(13) & _
            "Would you like to enter values into the highlighted mandatory cells?" & Chr(13) & _
            " ", vbYesNo)
        DoEvents
        If Response = vbYes Then
            Cancel = True
            Exit Sub
        End If
    End If
End If
End If

'hide worksheets
ThisWorkbook.Unprotect ("fpsheets")
Sheets("FP template").Visible = False
'Sheets("Dropdown Lists").Visible = False
ThisWorkbook.Protect Password:="fpsheets", Structure:=True, Windows:=False

'cancel the open ontime instance
Dim Macro As String
Dim NextOnTime_ As Date
Macro = "z_CheckWhetherToSetANewTime"
NextOnTime_ = NextOnTime
*****

'Call z_CancelNewOnTime(NextOnTime_, Macro)
*****

On Error GoTo UnexpectedError
'check in
AbortEvent_Workbook_BeforeClose = True
*****

'Call z_CheckIn("BeforeCloseCheckIn")
*****

'if workbook not saved ask what to do
If ThisWorkbook.Saved = False Then
    'replacement of the Excel MsgBox after closing of non saved workbooks

```

```

    Select Case MsgBox("Would you like to save the changes in the document.", vbQuestion +
vbYesNoCancel, "VBA Message")
        'Do not close the workbook
        Case vbCancel 'vbcancel=2
            Cancel = True 'Excel MsgBox erscheint nicht
            AbortEvent_Workbook_BeforeClose = False
            'NotUsed = MsgBox("Document not saved.", vbOKOnly, "VBA Message")
            'unhide worksheets
            ThisWorkbook.Unprotect ("fpsheets")
            Sheets("FP template").Visible = True
            'Sheets("Dropdown Lists").Visible = True
            ThisWorkbook.Protect Password:="fpsheets", Structure:=True, Windows:=False
            Exit Sub
        'Save and close the workbook
        Case vbYes: ThisWorkbook.Save 'vbyes=6 save and close
        'Close the workbook without saving (set the saved property to true)
        Case vbNo: ThisWorkbook.Saved = True 'vbno=7 not save just close
    End Select
End If

On Error GoTo UnexpectedError
'check in
AbortEvent_Workbook_BeforeClose = True
'*****

Call z_CheckIn("BeforeCloseCheckIn")
'*****

On Error Resume Next
'Excel MsgBox should not occur (set the saved property to true)
If ThisWorkbook.Saved = False Then
    ThisWorkbook.Saved = True
End If
Exit Sub
UnexpectedError:
    ThisWorkbook.Close True
    Exit Sub
End Sub

Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
    On Error GoTo UnexpectedError
    If SaveAsUI = True Then
        'info
        NotUsed = MsgBox("Please do not save this workbook locally." & Chr(13) & _
        "In order to make a local copy you need to select the required cells" & Chr(13) & _
        "and then copy and paste them into a sheet within a new workbook." & _
        , vbInformation + vbOKOnly, "VBA Message")
        'do not save
        Cancel = True
    End If
Exit Sub
UnexpectedError:
    ThisWorkbook.Close True
    Exit Sub

```

End Sub

```
Function z_CheckWhetherToSetANewTime()  
    On Error GoTo UnexpectedError  
    Debug.Print "z_CheckWhetherToSetANewTime_b " & Now() & " " & LastTime & " " &  
LastInteraction & " " & LastOnTime & " " & NextOnTime  
    Dim Macro As String  
    Dim NextOnTime_ As Date  
  
    'workbook/laptop is connected to SP and not checked out automatically but manually after  
opening or _  
    'workbook/laptop is connected to SP and checked out automatically after opening  
    If Workbooks(ThisWorkbook.Name).CanCheckIn = True And Flag_WhenOpenNotCheckedOut =  
True Or _  
        Workbooks(ThisWorkbook.Name).CanCheckIn = True And Flag_WhenOpenNotCheckedOut =  
False Then  
        'there was an interaction: set a new ontime  
        If LastInteraction > LastTime Then  
            Debug.Print "z_CheckWhetherToSetANewTime_b LastInteraction>LastTime "  
            'reset  
            GoodBy_Flag = False  
            LastTime = LastOnTime  
            LastOnTime = NextOnTime  
            NextOnTime = NextOnTime + AcceptedDuration  
            Macro = "z_CheckWhetherToSetANewTime"  
            NextOnTime_ = NextOnTime  
            Call z_SetNewOnTime(NextOnTime_, Macro)  
            'there was no interaction: timeout  
        Else  
            'GoodBy_Flag still set false (after AcceptedDuration)  
            If GoodBy_Flag = False Then  
                Debug.Print "z_CheckWhetherToSetANewTime_b GoodByFlag=False "  
                'reset  
                GoodBy_Flag = True  
                LastTime = LastOnTime  
                LastOnTime = NextOnTime  
                NextOnTime = NextOnTime + TimeOutDuration  
                Macro = "z_CheckWhetherToSetANewTime"  
                NextOnTime_ = NextOnTime  
                Call z_SetNewOnTime(NextOnTime_, Macro)  
                Call z_TimeOut  
                'GoodBy_Flag set true (after TimeOutDuration)  
            Else  
                Debug.Print "z_CheckWhetherToSetANewTime_b GoodByFlag=True "  
                'check in  
                *****  
                'Call z_CheckIn("ForcedCheckIn")  
                *****  
            End If  
        End If  
    End If  
    'workbook/laptop is not connected to SP and not checked out automatically after opening
```

```

Elseif Workbooks(ThisWorkbook.Name).CanCheckIn = False And Flag_WhenOpenNotCheckedOut
= True Then
    'there was an interaction: set a new ontime
    If LastInteraction > LastTime Then
        Debug.Print "z_CheckWhetherToSetANewTime_b LastInteraction>LastTime "
        'reset
        GoodBy_Flag = False
        LastTime = LastOnTime
        LastOnTime = NextOnTime
        NextOnTime = NextOnTime + AcceptedDuration
        Macro = "z_CheckWhetherToSetANewTime"
        NextOnTime_ = NextOnTime
        Call z_SetNewOnTime(NextOnTime_, Macro)
        Debug.Print Now() & " interaction"
    'there was no interaction: set a new ontime and bring no message box
    Else
        Debug.Print "z_CheckWhetherToSetANewTime_b LastInteraction>LastTime "
        'reset
        GoodBy_Flag = False
        LastTime = LastOnTime
        LastOnTime = NextOnTime
        NextOnTime = NextOnTime + AcceptedDuration
        Macro = "z_CheckWhetherToSetANewTime"
        NextOnTime_ = NextOnTime
        Call z_SetNewOnTime(NextOnTime_, Macro)
        Debug.Print Now() & " no interaction"
    End If
    'workbook/laptop is not connected to SP and checked out automatically after opening
Elseif Workbooks(ThisWorkbook.Name).CanCheckIn = False And Flag_WhenOpenNotCheckedOut
= False Then
    'there was an interaction: set a new ontime
    If LastInteraction > LastTime Then
        Debug.Print "z_CheckWhetherToSetANewTime_b LastInteraction>LastTime "
        'reset
        GoodBy_Flag = False
        LastTime = LastOnTime
        LastOnTime = NextOnTime
        NextOnTime = NextOnTime + AcceptedDuration
        Macro = "z_CheckWhetherToSetANewTime"
        NextOnTime_ = NextOnTime
        Call z_SetNewOnTime(NextOnTime_, Macro)
    'there was no interaction: set a new ontime and bring a message box
    Else
        Debug.Print "z_CheckWhetherToSetANewTime_b LastInteraction>LastTime "
        'reset
        GoodBy_Flag = False
        LastTime = LastOnTime
        LastOnTime = NextOnTime
        NextOnTime = NextOnTime + AcceptedDuration
        Macro = "z_CheckWhetherToSetANewTime"
        NextOnTime_ = NextOnTime
        Call z_SetNewOnTime(NextOnTime_, Macro)
        'show info

```

```

        UserForm6.Show (vbModeless)
        DoEvents
        Application.Wait (Now() + TimeValue("00:00:05"))
        UserForm6.Hide
    End If
End If

    Debug.Print "z_CheckWhetherToSetANewTime_e " & Now() & " " & LastTime & " " &
LastInteraction & " " & LastOnTime & " " & NextOnTime
Exit Function
UnexpectedError:
    ThisWorkbook.Close True
End Function
Function z_SetNewOnTime(NextOnTime_ As Date, Macro As String)
    Debug.Print "z_SetNewOnTime_b " & Now() & " " & LastTime & " " & LastInteraction & " " &
LastOnTime & " " & NextOnTime
    'Set a new NextOnTime value and start a new OnTime instance
    Application.OnTime NextOnTime_, Macro, , True
End Function
Function z_CancelNewOnTime(NextOnTime_ As Date, Macro As String)
    Debug.Print "z_CancelNewOnTime_b " & Now() & " " & LastTime & " " & LastInteraction & " " &
LastOnTime & " " & NextOnTime
    'Cancel the new OnTime instance
    Application.OnTime NextOnTime_, Macro, , False
End Function
Function z_TimeOut()
    Debug.Print "z_TimeOut_b " & Now() & " " & LastTime & " " & LastInteraction & " " & LastOnTime
& " " & NextOnTime
    On Error Resume Next
    UserForm1.Hide
    On Error GoTo 0
    UserForm2.Show (vbModeless)
End Function
Function z_CheckIn(Reason As String)
    Debug.Print "z_CheckIn_b " & Now() & " " & LastTime & " " & LastInteraction & " " & LastOnTime & "
" & NextOnTime
    'Stop

    'hide user forms
    On Error Resume Next
        UserForm1.Hide
        UserForm2.Hide
        UserForm3.Hide
        UserForm4.Hide
        UserForm5.Hide
        UserForm6.Hide
    On Error GoTo 0

    'get the user name
    Dim User As String
    Dim UserMail As String
    Set Olk = CreateObject("Outlook.Application")
    On Error GoTo error_label:

```



```

    UserMail = Olk.Session.CurrentUser.AddressEntry.GetExchangeUser.PrimarySmtpAddress
    Set Olk = Nothing
On Error GoTo 0
User = getUsername1

'workbook/laptop is connected to SP
If Workbooks(ThisWorkbook.Name).CanCheckIn = True Then
    'show info
    UserForm3.Show (vbModeless)
    DoEvents
    Application.Wait (Now() + TimeValue("00:00:05"))
    UserForm3.Hide
    'suppress the Before_Close event (and Before_Save)
    AbortEvent_Workbook_BeforeClose = True
    'save, check in and close workbook (saves only if WB.saved not set to true?)
    Workbooks(ThisWorkbook.Name).CheckIn True, User & ", " & UserMail, True
'workbook/laptop is not connected to SP
Else
    'test whether this path is still possible!!!
    If Reason = "ForcedCheckIn" Then
        'not connected to the internet, set a NextOnTime
        'simulate a new interaction
        LastInteraction = Now()
        'reset
        Dim Macro As String
        Dim NextOnTime_ As Date
        GoodBy_Flag = False
        LastTime = LastOnTime
        LastOnTime = NextOnTime
        NextOnTime = NextOnTime + AcceptedDuration
        Macro = "z_CheckWhetherToSetANewTime"
        NextOnTime_ = NextOnTime
        Call z_SetNewOnTime(NextOnTime_, Macro)
    'not connected to the internet and the workbook that is checked out
    ElseIf Reason = "BeforeCloseCheckIn" And Flag_WhenOpendNotCheckedOut = False Then
        'UserForm4.Show (vbModeless)
        'DoEvents
        'Application.Wait (Now() + TimeValue("00:00:10"))
        'UserForm4.Hide
        UserForm4.Show (vbModal)
        DoEvents
    'not connected to the internet and not the workbook that is checked out
    ElseIf Reason = "BeforeCloseCheckIn" And Flag_WhenOpendNotCheckedOut = True Then
        'UserForm5.Show (vbModeless)
        'DoEvents
        'Application.Wait (Now() + TimeValue("00:00:07"))
        'UserForm5.Hide
        UserForm5.Show (vbModal)
        DoEvents
    End If
End If
Debug.Print "z_CheckIn_e !!!!!!!!!!!this should be the last debug.print line!!!!!!!!!! "
Exit Function

```

```
error_label:
    UserMail = "Infos N/A"
    Resume Next
End Function
```

```
Sub SwitchOff_EnableEvents()
    Application.EnableEvents = False
End Sub
```

```
Sub SwitchOn_EnableEvents()
    Application.EnableEvents = True
End Sub
```

```
Sub GenerateTemplate()
    Call SwitchOff_EnableEvents
    Call UnprotectWorkbook
    Call UnhideSheets
    Call UnprotectSheets
    '---
    Call prepareSheetWithInputDropDownLists
    Call CopyPasteSheetWithInputDropdownLists
    Call RowHeightAndColumnWidth
    Call formatTemplate
    Call SetNumberFormats_DropdownList
    Call AddFormula_LinkToDropdownList
    Call SetNumberFormats_Template
    Call DataValidation_List
    Call addUniqueIdentifier
    Call ChangeErrorRules
    Call addAutofilter
    Call freezePane
    Call ManualSteps
    Call addEditableRange_ForTemplate_And_DataSetBackup
    Call Create_DataSetBackup_And_DataSetCheckOutVersion
    '---
    Call ProtectSheets
    Call ProtectWorkbook
    Call SwitchOn_EnableEvents
End Sub
Private Sub SwitchOff_EnableEvents()
    Application.EnableEvents = False
End Sub
```

```
Private Sub UnprotectWorkbook()
    'unprotect the workbook (necessary to unhide sheets)
    ThisWorkbook.Unprotect ("fpsheets")
End Sub
```

```
Private Sub UnhideSheets()
```

```

'unhide worksheets
Sheets("FP template").Visible = True
Sheets("Dropdown Lists").Visible = True
Sheets("DataSetBackup").Visible = True 'will be unhidden again below after copying
Sheets("UserNames").Visible = True 'will be unhidden again below after copying
Sheets("DataSetCheckOutVersion").Visible = True 'will be unhidden again below after copying
End Sub

```

```

Private Sub UnprotectSheets()
'unprotect all worksheets and vba
Sheets("FP template").Activate
ActiveSheet.Unprotect Password:="fpsheets"
Sheets("DataSetBackup").Activate
ActiveSheet.Unprotect Password:="fpsheets"
Sheets("Dropdown Lists").Activate
ActiveSheet.Unprotect Password:="fpsheets"
End Sub

```

```

Private Sub prepareSheetWithInputDropDownLists()
    answer = InputBox("Click OK and open the other workbook and activate dropdown template sheet", , "OK", 13500, 12500)
    answer = InputBox("The dropdown template sheet should have a consistent formatting: some wrong formats are inherited to the backup file and from there to the template", , "OK", 13500, 12500)
    Stop
    Rows("5:1099").Select
    'height
    Selection.RowHeight = 15
    'no font color
    With Selection.Font
        .ColorIndex = xlAutomatic
        .TintAndShade = 0
    End With
    'no interior color
    With Selection.Interior
        .Pattern = xlNone
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
    'no bold
    Selection.Font.Bold = True
    Selection.Font.Bold = False
    'alignement vertical
    With Selection
        .VerticalAlignment = xlTop
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    With Selection

```

```

        .VerticalAlignment = xlBottom
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    'alignement horizontal
    With Selection
        .HorizontalAlignment = xlRight
        .VerticalAlignment = xlBottom
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    With Selection
        .HorizontalAlignment = xlLeft
        .VerticalAlignment = xlBottom
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    'no wrap
    With Selection
        .HorizontalAlignment = xlLeft
        .VerticalAlignment = xlBottom
        .WrapText = True
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    With Selection
        .HorizontalAlignment = xlLeft
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    'no italic

```

```

Selection.Font.Italic = True
Selection.Font.Italic = False
'no underline
Selection.Font.Underline = xlUnderlineStyleSingle
Selection.Font.Underline = xlUnderlineStyleNone
'activate this workbook again
ThisWorkbook.Activate
End Sub
Private Sub CopyPasteSheetWithInputDropdownLists()
'to
Dim wb_to As Workbook
Set wb_to = ThisWorkbook
'to 1
Dim Sh_to1 As String
Sh_to1 = "Dropdown Lists"
Dim Cell_to1 As Range
Set Cell_to1 = Sheets(Sh_to1).Cells(1, 1)
'to 2
Dim Sh_to2 As String
Sh_to2 = "FP template"
Dim Cell_to2 As Range
Set Cell_to2 = Sheets(Sh_to2).Cells(20000, 1)
'to 3
Dim Sh_to3 As String
Sh_to3 = "FP template"
Dim Cell_to3 As Range
Set Cell_to3 = Sheets(Sh_to3).Cells(1, 1)
'from
answer = InputBox("Click OK and open the other workbook and activate dropdown template
sheet", , "OK", 13500, 12500)
Stop
Dim Sh_from As String
Dim wb_from As Workbook
Set wb_from = ActiveWorkbook
Sh_from = ActiveSheet.Name
'range
Dim Rng_Cpy As Range
Dim ColSize As Long
ColSize = z_ColSize(1, Sh_from)
Set Rng_Cpy = Sheets(Sh_from).Range(Sheets(Sh_from).Cells(1, 1), Sheets(Sh_from).Cells(1000,
ColSize))
'paste 1
Call z_CopyRange3("All", Sh_from, Rng_Cpy, Sh_to1, Cell_to1, wb_from, wb_to)
'paste 2
Call z_CopyRange3("All", Sh_from, Rng_Cpy, Sh_to2, Cell_to2, wb_from, wb_to)
'paste 3
Call z_CopyRange3("All", Sh_from, Rng_Cpy, Sh_to3, Cell_to3, wb_from, wb_to)
End Sub

Private Sub RowHeightAndColumnWidth()
Dim Sh_template As String
Dim Sh_Dd As String
Sh_template = "FP template"

```

```

Sheets(Sh_template).Activate
ColSize = z_ColSize(1, Sh_template)
Sh_Dd = "Dropdown Lists"
'Width Template and Dropdown
Dim Col As Long
For Col = 1 To ColSize
    Dim String_in As String
    Dim String_out As String
    Dim String_arr As Variant
    Dim Width_ As Double
    String_in = Right(Cells(3, Col), 15)
    String_out = z_FindAndReplaceInString(String_in, _
        Array("[", "]"), Array("@", "@"))
    String_arr = Split(String_out, "@")
    If UBound(String_arr) > 1 Then
        Width_ = String_arr(UBound(String_arr) - 1)
        Call z_ChangeColWidth(Width_, Sh_template, Col, Col)
        Call z_ChangeColWidth(Width_, Sh_Dd, Col, Col)
    End If
Next
'Height Template
Call z_ChangeRowHeight(60, Sh_template, 1, 1)
Call z_ChangeRowHeight(64.5, Sh_template, 2, 2)
Call z_ChangeRowHeight(30, Sh_template, 3, 3)
Call z_ChangeRowHeight(36, Sh_template, 4, 4)
Call z_ChangeRowHeight(15, Sh_template, 5, 10000)
'Height Dropdown
Call z_ChangeRowHeight(60, Sh_Dd, 1, 1)
Call z_ChangeRowHeight(64.5, Sh_Dd, 2, 2)
Call z_ChangeRowHeight(42, Sh_Dd, 3, 3)
Call z_ChangeRowHeight(36, Sh_Dd, 4, 4)

End Sub

Private Sub formatTemplate()
    Dim Sh As String
    Sh = "FP template"
    Sheets(Sh).Activate
    ColSize = z_ColSize(1, Sh)

    'Clear
    Range(Cells(5, 1), Cells(10000, ColSize)).Select
    Selection.ClearContents
    'Border
    Dim Rng As Range
    Set Rng = Union(Range(Cells(5, 1), Cells(10000, ColSize)), Range(Cells(20004, 1), Cells(21099,
ColSize)))
    Rng.Select
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = 0
    End With

```

```

        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideHorizontal)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
End Sub

Private Sub SetNumberFormats_DropdownList()
    Dim Sh As String
    Sh = "Dropdown Lists"
    Sheets(Sh).Activate
    ColSize = z_ColSize(1, Sh)

    ColStart = 1

    For Col = ColStart To ColSize
        Dim String_in As String
        Dim String_out As String
        Dim String_arr As Variant
        Dim Format_ As String
        String_in = Right(Cells(3, Col), 10)
        String_out = z_FindAndReplaceInString(String_in, _
            Array("(", ")", " "), Array("@", "@"))
        String_arr = Split(String_out, "@")
        Format_ = String_arr(UBound(String_arr) - 1)
    
```

```

Cells(3, Col).Select
'Dim answer As String
'answer = InputBox("Column " & CStr(Col) & " has Number-, Date- or Text-format", , Format_,
13500, 12500)
If LCase(Format_) = "number" Then
    Range(Cells(5, Col), Cells(10000, Col)).Select
    Selection.NumberFormat = "#,##0"
ElseIf LCase(Format_) = "date" Then
    Range(Cells(5, Col), Cells(10000, Col)).Select
    Selection.NumberFormat = "dd/mm/yyyy;@"
ElseIf LCase(Format_) = "text" Then
    Range(Cells(5, Col), Cells(10000, Col)).Select
    Selection.NumberFormat = "@"
Else
    Stop
    Range(Cells(5, Col), Cells(10000, Col)).Select
    Selection.NumberFormat = "General"
End If
Next
End Sub

```

```

Private Sub AddFormula_LinkToDropdownList()
    Dim Sh As String
    Dim Sh_from As String
    Sh = "FP template"
    Sheets(Sh).Activate
    ColSize = z_ColSize(1, Sh)
    Sh_from = "Dropdown Lists"

    Delta = 19999
    For Row = 20004 To 21099
        For Col = 1 To ColSize
            Dim Address_R1C1 As String
            Address_R1C1 = "R" & Row - Delta & "C" & Col
            Address_A1 = z_Converter_R1C1vsA1(Address_R1C1, "xlA1", Cells(1, 1))
            'Cells(Row, Col).Value = "=" & Sh_from & "!" & Address_A1
            Cells(Row, Col).NumberFormat = "General"
            Cells(Row, Col).Formula = "=" & Sh_from & "!" & Address_A1
        Next
    Next
End Sub

```

```

Private Sub SetNumberFormats_Template()
    Dim Sh As String
    Sh = "FP template"
    Sheets(Sh).Activate
    ColSize = z_ColSize(1, Sh)

    ColStart = 1

    For Col = ColStart To ColSize
        Dim String_in As String
        Dim String_out As String
    Next
End Sub

```



```

Dim String_arr As Variant
Dim Format_ As String
String_in = Right(Cells(3, Col), 10)
String_out = z_FindAndReplaceInString(String_in, _
    Array("(", ")", Array("@", "@"))
String_arr = Split(String_out, "@")
Format_ = String_arr(UBound(String_arr) - 1)
Cells(3, Col).Select
'Dim answer As String
'answer = InputBox("Column " & CStr(Col) & " has Number-, Date- or Text-format", , Format_,
13500, 12500)
If LCase(Format_) = "number" Then
    Range(Cells(6, Col), Cells(10000, Col)).Select
    Selection.NumberFormat = "#,##0"
    Range(Cells(20004, Col), Cells(21100, Col)).Select
    Selection.NumberFormat = "#,##0"
Elseif LCase(Format_) = "date" Then
    Range(Cells(6, Col), Cells(10000, Col)).Select
    Selection.NumberFormat = "dd/mm/yyyy;@"
    Range(Cells(20004, Col), Cells(21100, Col)).Select
    Selection.NumberFormat = "dd/mm/yyyy;@"
Elseif LCase(Format_) = "text" Then
    Range(Cells(6, Col), Cells(10000, Col)).Select
    Selection.NumberFormat = "@"
    Range(Cells(20004, Col), Cells(21100, Col)).Select
    Selection.NumberFormat = "@"
Else
    Stop
    Range(Cells(6, Col), Cells(10000, Col)).Select
    Selection.NumberFormat = "General"
    Range(Cells(20004, Col), Cells(21100, Col)).Select
    Selection.NumberFormat = "General"
End If
Next
End Sub

```

```

Private Sub DataValidation_List()
Dim Sh As String
Sh = "FP template"
Sheets(Sh).Activate
ColSize = z_ColSize(1, Sh)
ColStart = 2
Dim Col As Long
For Col = ColStart To ColSize
    If Cells(4, Col).Value Like "*drop down*" Then
        RowSize = z_RowSize(Col, Sh)
        For Row = RowSize To 20004 Step -1
            If Cells(Row, Col) <> Empty Then
                LastRow = Row
                Exit For
            End If
        Next
        Dim AddressFirst As String
    End If
Next
End Sub

```

```

Dim AddressLast As String
Dim DropdownAddress As String
AddressFirst = Cells(20004, Col).Address
AddressLast = Cells(LastRow, Col).Address
DropdownAddress = "=" & AddressFirst & ":" & AddressLast
Range(Cells(6, Col), Cells(10000, Col)).Select
'answer = InputBox("Dropdownlist " & DropdownAddress, , "OK", 13500, 12500)
'If answer <> "OK" Then
'    Stop
'End If
With Selection.Validation
    .Delete
    .Add Type:=xlValidateList, AlertStyle:=xlValidAlertStop, Operator:= _
        xlBetween, Formula1:=DropdownAddress
    .IgnoreBlank = True
    .InCellDropdown = True
    .InputTitle = ""
    .ErrorTitle = ""
    .InputMessage = ""
    .ErrorMessage = ""
    .ShowInput = True
    .ShowError = True
End With
End If
Next
End Sub

```

```

Private Sub addUniqueIdentifier()
Dim Sh As String
Sh = "FP template"
Sheets(Sh).Activate
answer = InputBox("Enter the first digit of the identifier", , 1, 13500, 12500)
For Row = 6 To 10000
    Cells(Row, 1).Value = "" & CStr(answer) & CStr(Right("00000" & CStr(Row - 5), 4))
Next
Range(Cells(5, 1), Cells(10000, 1)).Select
With Selection
    .HorizontalAlignment = xlCenter
    .VerticalAlignment = xlBottom
    .WrapText = False
    .Orientation = 0
    .AddIndent = False
    .IndentLevel = 0
    .ShrinkToFit = False
    .ReadingOrder = xlContext
    .MergeCells = False
End With
Selection.Interior.Color = RGB(218, 238, 243)
End Sub

```

```

Private Sub ChangeErrorRules()
Dim Sh As String
Sh = "FP template"

```

```

Sheets(Sh).Activate
With Application.ErrorCheckingOptions
    .BackgroundChecking = True
    .EvaluateToError = True
    .InconsistentFormula = True
    .NumberAsText = False
End With
End Sub

```

```

Private Sub addAutofilter()
    Dim Sh As String
    Sh = "FP template"
    Sheets(Sh).Activate
    'off
    If Sheets(Sh).AutoFilterMode = True Then
        Selection.AutoFilter
    End If
    'on
    Range("A4:AH4").Select
    Selection.AutoFilter
End Sub

```

```

Private Sub freezePane()
    Dim Sh As String
    Sh = "FP template"
    Sheets(Sh).Activate
    'unfreeze
    ActiveWindow.SplitColumn = 0
    ActiveWindow.SplitRow = 0
    'freeze
    Range("D5").Select
    ActiveWindow.FreezePanels = True
End Sub

```

```

Private Sub ManualSteps()
    MsgBox ("Beispielszeile in row 5")
    Stop
End Sub

```

```

Private Sub addEditableRange_ForTemplate_And_DataSetBackup()
    Dim Sh_template As String
    Dim Sh_bkup As String
    Sh_template = "FP template"
    Sh_bkup = "DataSetBackup"

    Dim ColSize As Long
    Sheets(Sh_template).Activate
    ColSize = z_ColSize(1, Sh_template)
    'EditRange1
    Sheets(Sh_template).Protection.AllowEditRanges.Add Title:="Range1_template",
Range:=Range(Cells(6, 2), Cells(10000, ColSize))
    'Edit Range2

```

```

    Sheets(Sh_bkup).Protection.AllowEditRanges.Add Title:="Range1_bkup", Range:=Range(Cells(6, 2),
Cells(10000, ColSize))
End Sub

```

```

Private Sub Create_DataSetBackup_And_DataSetCheckOutVersion()

```

```

    Dim Sh As String
    Dim Sh_bkup As String
    Dim Sh_chkout As String
    Sh = "FP template"
    Sh_bkup = "DataSetBackup"
    Sh_chkout = "DataSetCheckOutVersion"

```

```

    Sheets(Sh).Activate
    ColSize = z_ColSize(1, Sh)

```

```

    Dim Rng_Cpy As Range
    Set Rng_Cpy = Sheets(Sh).Range(Sheets(Sh).Cells(1, 1), Sheets(Sh).Cells(10000, ColSize))
    Dim Cell_to As Range
    'paste1
    Set Cell_to = Sheets(Sh_bkup).Cells(1, 1)
    Call z_CopyRange("All", Sh, Rng_Cpy, Sh_bkup, Cell_to)
    'paste2
    Set Cell_to = Sheets(Sh_chkout).Cells(1, 1)
    Call z_CopyRange("All", Sh, Rng_Cpy, Sh_chkout, Cell_to)

```

```

End Sub

```

```

Private Sub ProtectSheets()

```

```

    'protect all worksheets and vba
    Sheets("FP template").Activate
    Cells.Select
    Selection.Locked = True
    ActiveSheet.Protect Password:="fpsheets", DrawingObjects:=True, Contents:=True,

```

```

    Scenarios:=True, _

```

```

        AllowFormattingCells:=True, AllowFormattingColumns:=True, _
        AllowFormattingRows:=True, AllowSorting:=True, AllowFiltering:=True, _
        AllowUsingPivotTables:=True

```

```

    Sheets("DataSetBackup").Activate

```

```

    ActiveSheet.Protect Password:="fpsheets", DrawingObjects:=True, Contents:=True,

```

```

    Scenarios:=True, _

```

```

        AllowFormattingCells:=True, AllowFormattingColumns:=True, _
        AllowFormattingRows:=True, AllowSorting:=True, AllowFiltering:=True, _
        AllowUsingPivotTables:=True

```

```

    Sheets("Dropdown Lists").Activate

```

```

    Cells.Select

```

```

    Selection.Locked = True

```

```

    ActiveSheet.Protect Password:="fpsheets", DrawingObjects:=True, Contents:=True,

```

```

    Scenarios:=True, _

```

```

        AllowFormattingCells:=True, AllowFormattingColumns:=True, _
        AllowFormattingRows:=True, AllowSorting:=True, AllowFiltering:=True, _
        AllowUsingPivotTables:=True

```

```

    ActiveSheet.Cells(1, 1).Select

```

```

End Sub

```

```

Private Sub ProtectWorkbook()
    'protect workbook again
    ThisWorkbook.Protect Password:="fpsheets", Structure:=True, Windows:=False
End Sub

```

```

Sub SwitchOn_EnableEvents()
    Application.EnableEvents = True
End Sub

```

```

Public Wbk_teamspacePathAndName As String
Public GoodBy_Flag As Boolean
Public LastTime As Date
Public LastOnTime As Date
Public NextOnTime As Date
Public LastInteraction As Date
Public Const AcceptedDuration As String = "20:20:30" ""00:10:00"
Public Const TimeOutDuration As String = "00:01:15" ""00:02:00"
Public AbortEvent_Workbook_BeforeClose As Boolean
Public Flag_WhenOpendNotCheckedOut As Boolean
Public Flag_HideSheets As Boolean
Public AbortEvent_Worksheet_Change As Boolean
Public count_CloseAttempts As Integer
Public AbortMandatoryCells_InEvent_Workbook_BeforeClose As Boolean

```

```

Function z_InteriorColor_MandatoryData() As Boolean
    'flag
    Dim MandatoryCells_OK As Boolean
    MandatoryCells_OK = True

    '___
    Dim ColSize As Long
    ColSize = z_ColSize(1, "FP template")
    '___

    'when opened
    Dim LastWrittenRow_Open As Long
    LastWrittenRow_Open = z_LastWrittenRow("DataSetCheckOutVersion", 2, ColSize, 10000)
    'before close
    Dim LastWrittenRow_Close As Long
    LastWrittenRow_Close = z_LastWrittenRow("FP template", 2, ColSize, 10000)
    If LastWrittenRow_Close > LastWrittenRow_Open Then
        'Range
        Dim Rng As Range
        Sheets("FP template").Activate
        Set Rng = Sheets("FP template")._
            Range(Cells(LastWrittenRow_Open + 1, 2), Cells(LastWrittenRow_Close, ColSize))
        For Each cell In Rng
            'Mandatory columns

```

```

'      If Cell.Column = 7 Or Cell.Column = 8 Or Cell.Column = 9 Or Cell.Column = 10 Or Cell.Column =
11 Or _
'      Cell.Column = 12 Or Cell.Column = 15 Or Cell.Column = 16 Or Cell.Column = 17 Or
Cell.Column = 20 Or _
'      Cell.Column = 21 Or Cell.Column = 24 Or Cell.Column = 25 Or Cell.Column = 26 Or
Cell.Column = 29 Or _
'      Cell.Column = 30 Then
'      '--
      If LCase(Cells(3, cell.Column)) Like "*"mandatory*" Then
'      '--
          'No entry
          If cell = Empty Then
              cell.Interior.Color = 2600000
              MandatoryCells_OK = False
          End If
      End If
      Next cell
  End If
  z_InteriorColor_MandatoryData = MandatoryCells_OK
End Function

```

```

Sub m_Unhide_HiddenSheets()
    'unprotect workbook
    ThisWorkbook.Unprotect ("fpsheets")
    'unhide
    Flag_HideSheets = False
    Sheets("DataSetBackup").Visible = True
    Sheets("UserNames").Visible = True
    Sheets("DataSetCheckOutVersion").Visible = True
    'protect workbook
    ThisWorkbook.Protect Password:="fpsheets", Structure:=True, Windows:=False
End Sub

```

```

Sub m_Hide_UnhiddenSheets()
    'unprotect workbook
    ThisWorkbook.Unprotect ("fpsheets")
    'unhide
    Flag_HideSheets = False
    Sheets("DataSetBackup").Visible = False
    Sheets("UserNames").Visible = False
    Sheets("DataSetCheckOutVersion").Visible = False
    Flag_HideSheets = True
    'protect workbook
    ThisWorkbook.Protect Password:="fpsheets", Structure:=True, Windows:=False
End Sub

```

```

Sub m_RemoveEmptyRows()
    'unprotect workbook
    ThisWorkbook.Unprotect ("fpsheets")
    'unhide
    Flag_HideSheets = False
    Sheets("DataSetBackup").Visible = True
    'unprotect sheets (they will be protected again when opened next time

```

```

Sheets("DataSetBackup").Activate
ActiveSheet.Unprotect ("fpsheets")
Sheets("FP template").Activate
ActiveSheet.Unprotect ("fpsheets")
'switch off enable events
Call z_SwitchOff_EnableEvents
'trim the whole range and remove empty rows
Call z_RemoveEmptyRows("FP template")

'__
Dim ColSize As Long
ColSize = z_ColSize(1, "FP template")
'__

'copy/paste into DataSetBackup
Sheets("FP template").Activate
Call z_CopyRange("Values", "FP template", Sheets("FP template").Range(Cells(1, 1), Cells(10000,
ColSize)), _
    "DataSetBackup", Sheets("DataSetBackup").Cells(1, 1))
'hide
Flag_HideSheets = True
Sheets("DataSetBackup").Visible = False
'save
ThisWorkbook.Save
'switch on enable events
Call z_SwitchOn_EnableEvents
'close workbook
ThisWorkbook.Close False
End Sub

```

```

Sub m_CopyTemplateIntoNewWorkbook()
    Dim Wb_template As Workbook
    Dim Wb_new As Workbook
    Set Wb_template = ThisWorkbook
    Set Wb_new = Workbooks.Add
    Wb_template.Activate
    Sheets("FP template").Activate
    Cells.Select
    Selection.Copy
    Wb_new.Activate
    Sheets(1).Select
    Cells.Select
    Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Range("A1").Select
    Sheets(1).Rows(1).RowHeight = 60
    Sheets(1).Rows(2).RowHeight = 64.5
    Sheets(1).Rows(3).RowHeight = 30
    Sheets(1).Rows(4).RowHeight = 36
End Sub

```

```

Function z_SwitchOff_EnableEvents()
    Application.EnableEvents = False

```

End Function

```
Function z_SwitchOn_EnableEvents()  
    Application.EnableEvents = True  
End Function
```

```
Function z_RemoveEmptyRows(Sh As String)  
    Dim CheckRow As Long  
    CheckRow = 6  
  
    '_____  
    Dim ColSize As Long  
    ColSize = z_ColSize(1, Sh)  
    '_____  
  
    Dim LastWrittenRow As Long  
    LastWrittenRow = z_LastWrittenRow(Sh, 2, ColSize, 10000)  
    For Iter = 6 To LastWrittenRow  
        Call z_TrimRow(Sh, 2, ColSize, CheckRow)  
        Dim IsEmptyRow As Boolean  
        IsEmptyRow = z_FindEmptyRow(Sh, 2, ColSize, CheckRow)  
        If IsEmptyRow = True Then  
            'Delete Cells between Col_From and Col_To  
            Call z_DeleteAndInsertRowRange("Sheet1", CheckRow, 15000, 2, ColSize)  
            'one row back to remain on the same row after the next statement  
            CheckRow = CheckRow - 1  
        End If  
        CheckRow = CheckRow + 1  
    Next Iter  
    'format  
    Call z_formats(Sh, 6, 2, LastWrittenRow, ColSize, False)  
End Function
```

```
Function z_DeleteAndInsertRowRange(Sh As String, RowDel As Long, RowIns As Long, _  
    Col_From As Long, Col_To As Long)  
    'Delete Row Range and shift up  
    Range(Cells(RowDel, Col_From), Cells(RowDel, Col_To)).Select  
    Selection.Delete Shift:=xlUp  
    'Insert Row Range and shift down  
    Range(Cells(RowIns, Col_From), Cells(RowIns, Col_To)).Select  
    Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove  
End Function
```

```
Function z_TrimRow(Sh As String, Col_From As Long, Col_To As Long, Optional CheckRow As Long)  
    Sheets(Sh).Activate  
    Dim Rng As Range  
    If CheckRow = 0 Then  
        Cells.Select  
        Set Rng = Selection.Cells  
    Else  
        Set Rng = Range(Cells(CheckRow, Col_From), Cells(CheckRow, Col_To))  
    End If
```



```

For Each cell In Rng
    cell.Value = Excel.WorksheetFunction.Trim(WorksheetFunction.Clean(cell))
Next cell
End Function

```

```

Function z_FindEmptyRow(Sh As String, Col_From As Long, Col_To As Long, CheckRow As Long) As
Boolean
    Sheets(Sh).Activate
    Dim Flag_IsEmpty As Boolean
    Dim Rng As Range
    Set Rng = Range(Cells(CheckRow, Col_From), Cells(CheckRow, Col_To))
    For Each cell In Rng
        If cell.Value = Empty Then
            Flag_IsEmpty = True
        Else
            Flag_IsEmpty = False
            z_FindEmptyRow = Flag_IsEmpty
            Exit Function
        End If
    Next cell
    z_FindEmptyRow = Flag_IsEmpty
End Function

```

```

Function z_FormatCellNumbers(Sh As String, NextEmptyRow As Long)
    On Error GoTo UnexpectedError
    'Switch off worksheet change event
    AbortEvent_Worksheet_Change = True

    '___
    'do the format change
    Sheets(Sh).Activate
    ColSize = z_ColSize(1, Sh)
    ColStart = 1
    For Col = ColStart To ColSize
        'get the format from template row 3
        Dim String_in As String
        Dim String_out As String
        Dim String_arr As Variant
        Dim Format_ As String
        String_in = Right(Cells(3, Col), 10)
        String_out = z_FindAndReplaceInString(String_in, _
            Array("(", ")", " "), Array("@", "@"))
        String_arr = Split(String_out, "@")
        If UBound(String_arr) > 1 Then
            Format_ = String_arr(UBound(String_arr) - 1)
            'change the format in the range
            If LCase(Format_) = "number" Then
                Range(Cells(6, Col), Cells(NextEmptyRow, Col)).Select
                Selection.NumberFormat = "#,##0"
            ElseIf LCase(Format_) = "date" Then
                Range(Cells(6, Col), Cells(NextEmptyRow, Col)).Select
            End If
        End If
    Next Col
    '___
    'switch on worksheet change event
    AbortEvent_Worksheet_Change = False
    '___
    'do the format change
    Sheets(Sh).Activate
    ColSize = z_ColSize(1, Sh)
    ColStart = 1
    For Col = ColStart To ColSize
        'get the format from template row 3
        Dim String_in As String
        Dim String_out As String
        Dim String_arr As Variant
        Dim Format_ As String
        String_in = Right(Cells(3, Col), 10)
        String_out = z_FindAndReplaceInString(String_in, _
            Array("(", ")", " "), Array("@", "@"))
        String_arr = Split(String_out, "@")
        If UBound(String_arr) > 1 Then
            Format_ = String_arr(UBound(String_arr) - 1)
            'change the format in the range
            If LCase(Format_) = "number" Then
                Range(Cells(6, Col), Cells(NextEmptyRow, Col)).Select
                Selection.NumberFormat = "#,##0"
            ElseIf LCase(Format_) = "date" Then
                Range(Cells(6, Col), Cells(NextEmptyRow, Col)).Select
            End If
        End If
    Next Col
    '___
    'switch on worksheet change event
    AbortEvent_Worksheet_Change = False
    '___
End Function

```

```

        Selection.NumberFormat = "dd/mm/yyyy;@"
    ElseIf LCase(Format_) = "text" Then
        Range(Cells(6, Col), Cells(NextEmptyRow, Col)).Select
        Selection.NumberFormat = "@"
    Else
        Stop
        Range(Cells(6, Col), Cells(NextEmptyRow, Col)).Select
        Selection.NumberFormat = "General"
    End If
End If
Next
'__

'Switch on worksheet change event
AbortEvent_Worksheet_Change = False
Exit Function
UnexpectedError:
    AbortEvent_Worksheet_Change = False
    ThisWorkbook.Close True
End Function

Function z_FormatCellNumbers_online(Sh_DataSet As Worksheet, Cell_DataSet As Range)
    On Error GoTo UnexpectedError
    'Switch off worksheet change event
    AbortEvent_Worksheet_Change = True

    '___
    'get the format from template row 3
    Dim String_in As String
    Dim String_out As String
    Dim Col As Long
    Dim String_arr As Variant
    Dim Format_ As String
    Col = Cell_DataSet.Column
    String_in = Right(Cells(3, Col), 10)
    String_out = z_FindAndReplaceInString(String_in, _
        Array("(" , ")"), Array("@", "@"))
    String_arr = Split(String_out, "@")
    Format_ = String_arr(UBound(String_arr) - 1)
    'change the format in the range
    If LCase(Format_) = "number" Then
        Cell_DataSet.NumberFormat = "#,##0"
    ElseIf LCase(Format_) = "date" Then
        Cell_DataSet.NumberFormat = "dd/mm/yyyy;@"
    ElseIf LCase(Format_) = "text" Then
        Cell_DataSet.NumberFormat = "@"
    Else
        Stop
        Cell_DataSet.NumberFormat = "General"
    End If
    '___

    'Switch on worksheet change event

```

```

    AbortEvent_Worksheet_Change = False
Exit Function
UnexpectedError:
    AbortEvent_Worksheet_Change = False
    ThisWorkbook.Close True
End Function

```

```

'Function z_FormatCellNumbers(Sh As String, NextEmptyRow As Long)
' On Error GoTo UnexpectedError
' 'Switch off worksheet change event
' AbortEvent_Worksheet_Change = True
' 'Change format to text: SPIRIT trial ID - customer
' Range(Cells(6, 2), Cells(NextEmptyRow, 8)).NumberFormat = "@"
' 'Change format to text: PLC / stage - PLC / stage
' Range(Cells(6, 9), Cells(NextEmptyRow, 9)).NumberFormat = "#,##0" '"" for general/standard
' 'Change format to text: requestor - unit of detailed facility
' Range(Cells(6, 10), Cells(NextEmptyRow, 19)).NumberFormat = "@"
' 'Change format to numbers: total no. plots (or no. detailed facility ex column S) - total sqm
(calculated)
' Range(Cells(6, 20), Cells(NextEmptyRow, 22)).NumberFormat = "#,##0"
' 'Change format to text: trial design - starting year (YYYY)
' Range(Cells(6, 23), Cells(NextEmptyRow, 24)).NumberFormat = "@"
' 'Change format to date: sowing time (from DD.MM.YYYY) - end of trial (to DD.MM.YYYY)
' Range(Cells(6, 25), Cells(NextEmptyRow, 30)).NumberFormat = "dd/mm/yyyy;@"
' 'Change format to numbers: total fee to farmer (EUR) - total fee to other 3rd party (EUR)
' Range(Cells(6, 31), Cells(NextEmptyRow, 32)).NumberFormat = "#,##0"
' 'Change format to text: remarks - remarks
' Range(Cells(6, 33), Cells(NextEmptyRow, 33)).NumberFormat = "@"
' 'Switch on worksheet change event
' AbortEvent_Worksheet_Change = False
'Exit Function
'UnexpectedError:
' AbortEvent_Worksheet_Change = False
' ThisWorkbook.Close True
'End Function

```

```

'Function z_FormatCellNumbers_online(Sh_DataSet As Worksheet, Cell_DataSet As Range)
' On Error GoTo UnexpectedError
' 'Switch off worksheet change event
' AbortEvent_Worksheet_Change = True
' 'Change format to text: SPIRIT trial ID - customer
' If Not Application.Intersect(Cell_DataSet, Range(Cells(Cell_DataSet.Row, 2),
Cells(Cell_DataSet.Row, 8))) Is Nothing Then
' Cell_DataSet.NumberFormat = "@"
' 'Change format to text: PLC / stage - PLC / stage
' ElseIf Not Application.Intersect(Cell_DataSet, Range(Cells(Cell_DataSet.Row, 9),
Cells(Cell_DataSet.Row, 9))) Is Nothing Then
' Cell_DataSet.NumberFormat = "#,##0" '"" for general/standard
' 'Change format to text: requestor - unit of detailed facility
' ElseIf Not Application.Intersect(Cell_DataSet, Range(Cells(Cell_DataSet.Row, 10),
Cells(Cell_DataSet.Row, 19))) Is Nothing Then
' Cell_DataSet.NumberFormat = "@"

```

```

' 'Change format to numbers: total no. plots (or no. detailed facility ex column S) - total sqm
(calculated)
' ElseIf Not Application.Intersect(Cell_DataSet, Range(Cells(Cell_DataSet.Row, 20),
Cells(Cell_DataSet.Row, 22))) Is Nothing Then
'   Cell_DataSet.NumberFormat = "#,##0"
' 'Change format to text: trial design - starting year (YYYY)
' ElseIf Not Application.Intersect(Cell_DataSet, Range(Cells(Cell_DataSet.Row, 23),
Cells(Cell_DataSet.Row, 24))) Is Nothing Then
'   Cell_DataSet.NumberFormat = "@"
' 'Change format to date: sowing time (from DD.MM.YYYY) - end of trial (to DD.MM.YYYY)
' ElseIf Not Application.Intersect(Cell_DataSet, Range(Cells(Cell_DataSet.Row, 25),
Cells(Cell_DataSet.Row, 30))) Is Nothing Then
'   Cell_DataSet.NumberFormat = "dd/mm/yyyy;@"
' 'Change format to numbers: total fee to farmer (EUR) - total fee to other 3rd party (EUR)
' ElseIf Not Application.Intersect(Cell_DataSet, Range(Cells(Cell_DataSet.Row, 31),
Cells(Cell_DataSet.Row, 32))) Is Nothing Then
'   Cell_DataSet.NumberFormat = "#,##0"
' 'Change format to text: remarks - remarks
' ElseIf Not Application.Intersect(Cell_DataSet, Range(Cells(Cell_DataSet.Row, 33),
Cells(Cell_DataSet.Row, 33))) Is Nothing Then
'   Cell_DataSet.NumberFormat = "@"
' End If
' Switch on worksheet change event
' AbortEvent_Worksheet_Change = False
'Exit Function
'UnexpectedError:
' AbortEvent_Worksheet_Change = False
' ThisWorkbook.Close True
'End Function

```

'formats

```

Function z_formats(Sh As String, Row_Rng_DataSet_first As Long, Col_Rng_DataSet_first As Long, _
    Row_Rng_DataSet_last As Long, Col_Rng_DataSet_last As Long, Flag_ChangeColor As Boolean)
On Error GoTo UnexpectedError:
Dim Rng_DataSet As Range
Sheets(Sh).Activate
Set Rng_DataSet = Sheets(Sh).Range(Cells(Row_Rng_DataSet_first, Col_Rng_DataSet_first), _
    Cells(Row_Rng_DataSet_last, Col_Rng_DataSet_last))

```

'If Row or column size has been changed

'Rng\_DataSet.RowHeight = 15

'Rng\_DataSet.ColumnWidth = 16

```

Rng_DataSet.Select
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With

```

```

With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
With Selection.Borders(xlInsideVertical)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
With Selection.Borders(xlInsideHorizontal)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
If Flag_ChangeColor = True Then
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .Color = 16316664
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
End If
With Selection.Font
    .ThemeColor = xlThemeColorLight1
    .TintAndShade = 0
End With
Exit Function
UnexpectedError:
    ThisWorkbook.Close True
End Function

Declare Function getUserNam Lib "advapi32.dll" Alias "GetUserNameA" (ByVal lpBuffer As String,
nSize As Long) As Long

Function z_SetCursor(Sh As String, StartAtCol As Long, StopAtCol As Long, StartAtRow As Long,
SetCursorAtCol As Long)

```

```

Dim SetCursorAtRow As Long
SetCursorAtRow = z_LastWrittenRow(Sh, StartAtCol, StopAtCol, StartAtRow)
Sheets(Sh).Activate
Sheets(Sh).Cells(SetCursorAtRow + 1, SetCursorAtCol).Select
End Function

```

```

Function z_LastWrittenRow(Optional Sh As String, Optional StartAtCol As Long, _
    Optional StopAtCol As Long, Optional StartAtRow As Long, Optional ByRef Wb As
Workbook) As Long

```

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 11.10.2011
'Input: All Columns, Output: last written row
'FirstEmptyCol = z_FirstEmptyCol()
Dim RowSize_max As Long: RowSize_max = 0
Dim RowSize_col As Long: RowSize_col = 0
Dim AllCols As Long: AllCols = 16384
Dim Col As Long
'Optional input
If StartAtCol = 0 Then
    StartAtCol = 1
End If
If StopAtCol = 0 Then
    StopAtCol = AllCols
End If
If StartAtRow = 0 Then
    StartAtRow = 1048576
End If
For Col = StartAtCol To StopAtCol
    RowSize_col = z_RowSize2(Col, Sh, StartAtRow)
    If RowSize_col > RowSize_max Then
        RowSize_max = RowSize_col
    End If
Next Col
z_LastWrittenRow = RowSize_max
End Function

```

```

Function z_RowSize2(SearchCol As Long, Optional Sh As String, Optional StartAtRow As Long,
Optional ByRef Wb As Workbook) As Long

```

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
If StartAtRow = 0 Then
    StartAtRow = 1048576
End If
z_RowSize2 = If(IsEmpty(Sheets(Sh).Cells(StartAtRow, SearchCol)), Sheets(Sh).Cells(StartAtRow,
SearchCol).End(xlUp).Row, StartAtRow)
End Function

```

```

Function getDesktopFolder() As String
Dim s As String
s = CreateObject("WScript.Shell").SpecialFolders("Desktop")
getDesktopFolder = s
End Function

```

```

Function getUsername2() As String
Dim s As String

```

```

Dim x As Variant
Dim User As String
s = CreateObject("WScript.Shell").SpecialFolders("Desktop")
x = Split(s, "\")
User = x(UBound(x) - 1)
getUserName2 = User
End Function

```

```

Function z_CopyRange(PasteAllOrValuesOrFormats As String, Sh_from As String, Range_from As
Range, _

```

```

    Sh_to As String, Cell_to As Range)
'newer version z_CopyRange2
'only works out if you invoke this function after this line!:Sheets(Sh_from).Activate
'otherwise VBA cannot set the Range_From
'set Cell_to as follows: Cell_to=Sheets(Sh_to).Range("A1")

```

```

Dim first As Long
Dim last As Long
If PasteAllOrValuesOrFormats = "All" Then
    Sheets(Sh_from).Activate
    first = z_Rng_firstCol(Range_from) 'Range_From.Columns.End(xlToLeft).Column
    last = z_Rng_lastCol(Range_from) 'Range_From.Columns.End(xlToRight).Column
    Call z_Copy_ColWidth(Sh_from, Sh_to, first, last)
    Sheets(Sh_from).Activate
    Range_from.Select
    Selection.Copy
    Sheets(Sh_to).Activate
    Cell_to.Select
    Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
ElseIf PasteAllOrValuesOrFormats = "Values" Then
    Sheets(Sh_from).Activate
    Range_from.Select
    Selection.Copy
    Sheets(Sh_to).Activate
    Cell_to.Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
ElseIf PasteAllOrValuesOrFormats = "Formats" Then
    Sheets(Sh_from).Activate
    Range_from.Select
    Selection.Copy
    Sheets(Sh_to).Activate
    Cell_to.Select
    Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
SkipBlanks:=False, Transpose:=False
End If
'Cells.Select
'Selection.RowHeight = 15
End Function

```

```

Function z_CopyRange3(PasteAllOrValuesOrFormats As String, Sh_from As String, Range_from As
Range, _

```

```

        Sh_to As String, Cell_to As Range, wb_from As Workbook, wb_to As Workbook)
'newer version z_CopyRange2
'only works out if you invoke this function after this line!:Sheets(Sh_from).Activate
'otherwise VBA cannot set the Range_From
'set Cell_to as follows: Cell_to=Sheets(Sh_to).Range("A1")

Dim first As Long
Dim last As Long
If PasteAllOrValuesOrFormats = "All" Then
    'copy
    wb_from.Activate
    Sheets(Sh_from).Activate
    first = z_Rng_firstCol(Range_from) 'Range_From.Columns.End(xlToLeft).Column
    last = z_Rng_lastCol(Range_from) 'Range_From.Columns.End(xlToRight).Column
    Call z_Copy_ColWidth(Sh_from, Sh_to, first, last, wb_from, wb_to)
    Sheets(Sh_from).Activate
    Range_from.Select
    Selection.Copy
    'paste
    wb_to.Activate
    Sheets(Sh_to).Activate
    Cell_to.Select
    Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks _
    :=False, Transpose:=False
ElseIf PasteAllOrValuesOrFormats = "Values" Then
    'copy
    wb_from.Activate
    Sheets(Sh_from).Activate
    Range_from.Select
    Selection.Copy
    'paste
    wb_to.Activate
    Sheets(Sh_to).Activate
    Cell_to.Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
    :=False, Transpose:=False
ElseIf PasteAllOrValuesOrFormats = "Formats" Then
    'copy
    wb_from.Activate
    Sheets(Sh_from).Activate
    Range_from.Select
    Selection.Copy
    'paste
    wb_to.Activate
    Sheets(Sh_to).Activate
    Cell_to.Select
    Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
End If
'Cells.Select
'Selection.RowHeight = 15
End Function

```



```
Function z_Rng_firstCol(ByRef Rng As Range) As Long
```

```
    z_Rng_firstCol = Rng(1, 1).Column
```

```
End Function
```

```
Function z_Rng_lastCol(ByRef Rng As Range) As Long
```

```
    Dim Rng_firstCol As Long
```

```
    Rng_firstCol = Rng(1, 1).Column
```

```
    z_Rng_lastCol = Rng_firstCol + Rng.Columns.Count - 1
```

```
End Function
```

```
Function z_Copy_ColWidth(Sh_from As String, Sh_to As String, Col_From As Long, Col_To As Long,  
Optional wb_from As Workbook, Optional wb_to As Workbook)
```

```
    If wb_from Is Nothing And wb_to Is Nothing Then
```

```
        For Col = Col_From To Col_To
```

```
            Sheets(Sh_to).Columns(Col).ColumnWidth = Sheets(Sh_from).Columns(Col).ColumnWidth
```

```
        Next Col
```

```
    Else
```

```
        For Col = Col_From To Col_To
```

```
            wb_to.Sheets(Sh_to).Columns(Col).ColumnWidth =
```

```
wb_from.Sheets(Sh_from).Columns(Col).ColumnWidth
```

```
        Next Col
```

```
    End If
```

```
End Function
```

```
Function z_ChangeColWidth(ColWidth As Double, Sh As String, Optional Col_From As Long, Optional  
Col_To As Long)
```

```
    If Col_From = 0 Then
```

```
        Col_From = 1
```

```
    End If
```

```
    If Col_To = 0 Then
```

```
        Col_To = 16384
```

```
    End If
```

```
    For Col = Col_From To Col_To
```

```
        If Sheets(Sh).Columns(Col).ColumnWidth <> ColWidth Then
```

```
            Sheets(Sh).Columns(Col).ColumnWidth = ColWidth
```

```
        End If
```

```
    Next Col
```

```
End Function
```

```
Function z_ChangeRowHeight(RowHeight As Double, Sh As String, Optional Row_From As Long,  
Optional Row_To As Long)
```

```
    If Row_From = 0 Then
```

```
        Row_From = 1
```

```
    End If
```

```
    If Row_To = 0 Then
```

```
        Row_To = 1048576
```

```
    End If
```

```
    For Row = Row_From To Row_To
```

```
        If Sheets(Sh).Rows(Row).RowHeight <> RowHeight Then
```

```
            Sheets(Sh).Rows(Row).RowHeight = RowHeight
```

```
        End If
```

```
    Next Row
```

```
End Function
```

```

Sub z_CopySheet(Sh_from As String, Sh_to As String)
    Sheets(Sh_from).Select
    Cells.Select
    Application.CutCopyMode = False
    Selection.Copy
    Sheets(Sh_to).Select
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _
        False, Transpose:=False
    Range("A1").Select
    Sheets(Sh_from).Select
    Range("A1").Select
End Sub

```

```

Function getUsername1() As String
    Dim Puffer As String * 256
    Dim User As String
    Dim Ret As Long
    Ret = getUsername(Puffer, Len(Puffer))
    If Ret <> 0 Then
        User = Left$(Puffer, InStr(1, Puffer, vbNullChar) - 1)
        getUsername1 = User
    End If
End Function

```

```

Function z_ShNewFlatValueCopy(Sh As String, Sh_new As String, Optional Where As String, Optional
ByRef Sh_Ref As Worksheet)

```

'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Name of the Sh to copy, Name of the new Sh\_new, where to place the new Sh\_new,  
' before or after some Sh\_Ref, at the begin or the end

```

    Call z_ShAdd(Where, Sh_Ref)
    On Error Resume Next
    ActiveSheet.Name = Sh_new
    If Err.Number <> 0 Then
        Application.DisplayAlerts = False
        ActiveSheet.Delete
        Application.DisplayAlerts = True
        Sheets(Sh_new).Cells.ClearContents
    End If
    On Error GoTo 0
    Sheets(Sh).Cells.Copy
    Dim WSh_new As Worksheet
    Set WSh_new = Sheets(Sh_new)
    WSh_new.Range("A1").PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    WSh_new.Columns("A:ZZ").ColumnWidth = 20
    'Sheets(Sh_new).Select
    'Sheets(Sh_new).Range("A1").Select
    'Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    'Sheets(Sh_new).Columns("A:GA").ColumnWidth = 20
End Function

```

```

Function z_ShAdd(Optional Where As String, Optional ByRef Sh_Ref As Worksheet)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: where to place the added Sh, before or after some Sh_Ref, at the begin or the end
If Not Sh_Ref Is Nothing Then
    If Where = ("Before:") Then
        Sheets.Add Sh_Ref
    ElseIf Where = ("After:") Then
        Sheets.Add , Sh_Ref
    Else
        Sheets.Add Before:=Sheets(1)
    End If
Else
    If Where = ("End") Then
        Sheets.Add After:=Sheets(Sheets.Count)
    ElseIf Where = ("Begin") Then
        Sheets.Add Before:=Sheets(1)
    Else
        Sheets.Add Before:=Sheets(1)
    End If
End If
End Function

```

```

Function z_LogUserName(Sh_UserNames As Worksheet, Cell_DataSet As Range)
    Dim Olk As Object
    Set Olk = CreateObject("Outlook.Application")
    Col = z_ColSize(Cell_DataSet.Row, Sh_UserNames.Name) + 1
    On Error GoTo label_error
    Sh_UserNames.Cells(Cell_DataSet.Row, Col) = _
        Olk.Session.CurrentUser.AddressEntry.GetExchangeUser.Name & ";" & _
        Olk.Session.CurrentUser.AddressEntry.GetExchangeUser.PrimarySmtpAddress & ";" & _
        Olk.Session.CurrentUser.AddressEntry.GetExchangeUser.Department & "; Time: " & _
        Now() & "; Column: " & _
        Cell_DataSet.Column & "; Value: " & _
        Cell_DataSet.Value
    Set Olk = Nothing
Exit Function
'if outlook is closed no user information can be determined
label_error:
    Dim User As String
    User = getUsername1
    Sh_UserNames.Cells(Cell_DataSet.Row, Col) = _
        User & _
        "Infos N/A" & "; Time: " & _
        Now() & "; Column: " & _
        Cell_DataSet.Column & "; Value: " & _
        Cell_DataSet.Value
End Function

```

```

Function z_CheckOut()
'Stop
    On Error GoTo Failed:

```

```

        'Workbook already open (since opened with clicking on the SP link) now check out without
opening
        Workbooks.CheckOut Wbk_teamspacePathAndName
        Flag_WhenOpendNotCheckedOut = False
        NotUsed = MsgBox(ThisWorkbook.Name & " has now been checked out automatically.",
vbOKOnly + vbInformation, "VBA Message")
        On Error GoTo 0
Exit Function
Failed:
        'this version is checked out
        If Workbooks(ThisWorkbook.Name).CanCheckIn = True Then
            NotUsed = MsgBox("This document has already been checked out manually.", vbOKOnly +
vbInformation, "VBA Message")
            'another version is checked out
        Else
            answer = MsgBox("Unable to check out this document at this time as likely checked out to other
user." & Chr(13) & _
            Chr(13) & _
            "Click Yes if you want to work offline." & Chr(13) & _
            "Click No if you want to close the document.", vbYesNo + vbQuestion, "VBA Message")
            If answer = vbYes Then
                'user may check in later manually??
                'set flag
                Flag_WhenOpendNotCheckedOut = True
                'disable events

                'not disable events

            ElseIf answer = vbNo Then
                'disable close event??? thats wrong
                'AbortEvent_Workbook_BeforeClose = True
                'enable close event
                AbortEvent_Workbook_BeforeClose = False
                AbortMandatoryCells_InEvent_Workbook_BeforeClose = True
                ThisWorkbook.Close (False)
            End If
        End If
End Function

'Sub test809()
' Call z_FormatCellNumbers("FP template")
' Call z_Insert_ProductFormula("U", "T", "W", "X", "V", 23, 25, _
'
'         "unique identifier", 1, "FP template")
'End Sub

Function z_Insert_ProductFormula(Col1 As String, Col2 As String, _
        Col3 As String, Col4 As String, Col5 As String, _
        ProdCol1 As Long, ProdCol2 As Long, _
        SearchColName As String, SearchRow As Integer, Sh As String)
'call z_Insert_ProductFormula("U", "T", "W", "X", "V", 23, 25, _
        "unique identifier", 1, "FP template")
'SearchColName=unique identifier

```

```

'ProdCol=23 total no. plants (calculated):=U6*T6
'ProdCol=25 total sqm (calculated):=IF(W6=0;T6*X6;IF(W6="";T6*X6;W6/V6))
'other possibility to enter the formula U6*T6 in W6 is:
  'ActiveCell.FormulaR1C1 = "=RC[-2]*RC[-3]"
On Error GoTo UnexpectedError
Dim SearchCol As Long
Dim RowSize As Long
Dim myformula1 As String
Dim myformula2 As String
Dim i As Long
Sheets(Sh).Select
SearchCol = z_GetColumnIndex(SearchColName, SearchRow, Sh)
RowSize = z_RowSize(SearchCol, Sh)

'Switch off worksheet change event
AbortEvent_Worksheet_Change = True
Application.EnableEvents = False

For i = 6 To RowSize
  'formula for ProdCol=23
  On Error GoTo ErrorInFormula
  If Cells(i, ProdCol1) = "" Then
    myformula1 = "=" & Col1 & CStr(i) & "*" & Col2 & CStr(i)
    'Cells(i, ProdCol1).ClearContents
    Cells(i, ProdCol1).Formula = myformula1
  End If

  On Error GoTo UnexpectedError
  'formula for ProdCol=25
  myformula2 = "=IF(" & Col3 & CStr(i) & "=0," & Col3 & CStr(i) & "*" & Col1 & CStr(i) & ",IF(" &
Col3 & CStr(i) & "=" & Chr(34) & Chr(34) & "," & Col2 & CStr(i) & "*" & Col4 & CStr(i) & "," & Col3 &
CStr(i) & "/" & Col5 & CStr(i) & "))"
  'Cells(i, ProdCol2).ClearContents
  Cells(i, ProdCol2).Formula = myformula2
Next i

'Switch on worksheet change event
AbortEvent_Worksheet_Change = False
Application.EnableEvents = True

Exit Function
ErrorInFormula:
  'msgbox: not a number in columns U and T
  Resume Next
UnexpectedError:
  AbortEvent_Worksheet_Change = False
  Application.EnableEvents = True
  ThisWorkbook.Close True
End Function

Public Function z_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, _
  Optional Sh As String, Optional ByRef Wb As Workbook) As Long
  'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

'Output datatype change from Variant
Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer
'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
'Sheets(Sh).Activate
Dim Sht As Worksheet
Set Sht = Sheets(Sh)
'find column name
Dim Cl As Range
'Set Cl = Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole)
Set Cl = Sht.Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole)
If Cl Is Nothing Then GoTo NameExpectedNotExistent
'find column index
'CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexStr = Cl.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
'Output
z_GetColumnIndex = ColIndex
Exit Function
OptionalArgument:
Resume Next
NameExpectedNotExistent:
ColIndex = 0
Stop 'only in test mode
z_GetColumnIndex = ColIndex
End Function

```

```

Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long

```

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

'Date: 26.9.2011

```

```

'Input: column, Output: row with the last entry in that column

```

```

'SearchCol datatype changed from integer

```

```

'Activate the Sheet

```

```

'Sheets(Sh).Activate

```

```

'Determine the row size

```

```

z_RowSize = If(IsEmpty(Sheets(Sh).Cells(1048576, SearchCol)), Sheets(Sh).Cells(1048576, SearchCol).End(xlUp).Row, 1048576)

```

```

End Function

```

```

Function z_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long

```

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

'Date: 26.9.2011

```

```

'Input: Row, Output: column with the last entry in that row

```

```

'SearchCol datatype changed from integer

```

```

'Activate the Sheet

```

```

'Sheets(Sh).Activate

```

```

'Determine the col size

```

```

    z_ColSize = If(IsEmpty(Sheets(Sh).Cells(SearchRow, 16384)), Sheets(Sh).Cells(SearchRow,
16384).End(xlToLeft).Column, 16384)

```

```

End Function

```

```

'Sub test6876()

```

```

,

```

```

' out = z_Converter_R1C1vsA1("A5", "xlR1C1", Cells(1, 1))

```

```

' out = z_Converter_R1C1vsA1("R40C20", "xl$A$1", Cells(1, 1))

```

```

' out = z_Converter_R1C1vsA1("R40C20", "xlA1", Cells(1, 1))

```

```

,

```

```

'End Sub

```

```

Function z_Converter_R1C1vsA1(Address As String, Optional ConversionType As String, Optional
RefCell As Range) As String

```

```

    Dim x As Variant

```

```

    If RefCell Is Nothing Then Set RefCell = ActiveCell

```

```

    If ConversionType = "xlR1C1" Then

```

```

        x = Application.ConvertFormula(Address, xlA1, xlR1C1, , RefCell) 'Convert A1 to R1C1

```

```

    Else

```

```

        x = Application.ConvertFormula(Address, xlR1C1, xlA1, , RefCell) 'Convert R1C1 to A1

```

```

        If ConversionType = "xlA1" Then

```

```

            x = Replace(x, "$", "")

```

```

        End If

```

```

    End If

```

```

    If IsError(x) Then

```

```

        z_Converter_R1C1vsA1 = Address

```

```

    Else

```

```

        'If input address is A1 reference and A1 is requested output, then Application.ConvertFormula
        'surrounds the address in single quotes.

```

```

        If Right(x, 1) = "" Then

```

```

            z_Converter_R1C1vsA1 = Mid(x, 2, Len(x) - 2)

```

```

        Else

```

```

            z_Converter_R1C1vsA1 = x

```

```

        End If

```

```

    End If

```

```

End Function

```

```

Function z_FindAndReplaceInString(String_ As String, What_ As Variant, Replacement_ As Variant, _
    Optional Start_ As Integer) As String

```

```

    Dim String_tmp As String

```

```

    String_tmp = String_

```

```

    If Start_ = Empty Then

```

```

        Dim What_i As String

```

```

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

```

```

            What_i = What_(i)

```

```

            Replacement_i = Replacement_(i)

```

```

            String_tmp = Replace(String_tmp, What_i, Replacement_i)

```

```

        Next

```

```

    Else

```

```

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

```

```

            What_i = What_(i)

```

```

        Replacement_i = Replacement_(i)
        String_tmp = Left(String_tmp, Start_ - 1) & Replace(String_tmp, What_i, Replacement_i,
Start_)
    Next
End If
z_FindAndReplaceInString = String_tmp
End Function

```

\*\*\*\*\*Get Range Indices

Function z\_RangeToIndices(ByRef Rng As Range) As Variant

'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

    Dim RangeIndices(0 To 3) As Long
    Dim CellsArray() As String
    Dim sAddr As String
    sAddr = Rng.Address(ReferenceStyle:=xlR1C1)
    CellsArray = Split(sAddr, ":")
    Dim CellIndicesUL() As Long
    On Error GoTo RangelsColumnOrRow
    CellIndicesUL = z_sCellToIndex(CellsArray(0))
    On Error GoTo 0
    Dim CellIndicesLR() As Long
    On Error GoTo RangelsCell
    CellIndicesLR = z_sCellToIndex(CellsArray(1))
    On Error GoTo 0
    RangeIndices(0) = CellIndicesUL(0)
    RangeIndices(1) = CellIndicesUL(1)
    RangeIndices(2) = CellIndicesLR(0)
    RangeIndices(3) = CellIndicesLR(1)
    z_RangeToIndices = RangeIndices

```

Exit Function

RangelsCell:

```
CellIndicesLR = z_sCellToIndex(CellsArray(0))
```

Resume Next

RangelsColumnOrRow:

```
Dim RorC As String
```

```
RorC = Left(sAddr, 1)
```

```
OneOrMore = InStr(1, sAddr, ":", vbTextCompare)
```

'only one row or column

```
If OneOrMore = 0 Then
```

```
    If RorC = "C" Then
```

```
        RangeIndices(0) = 1
```

```
        RangeIndices(1) = z_sColumnToIndex(CellsArray(0))
```

```
        RangeIndices(2) = 1048534
```

```
        RangeIndices(3) = RangeIndices(0)
```

```
    ElseIf RorC = "R" Then
```

```
        RangeIndices(0) = z_sRowToIndex(CellsArray(0))
```

```
        RangeIndices(1) = 1
```

```
        RangeIndices(2) = RangeIndices(0)
```

```
        RangeIndices(3) = 16383
```

```
    Else
```

```
        Stop
```

```
    End If
```

'more than one row or column



```

Else
    If RorC = "C" Then
        RangeIndices(0) = 1
        RangeIndices(1) = z_sColumnToIndex(CellsArray(0))
        RangeIndices(2) = 1048534
        RangeIndices(3) = z_sColumnToIndex(CellsArray(1))
    ElseIf RorC = "R" Then
        RangeIndices(0) = z_sRowToIndex(CellsArray(0))
        RangeIndices(1) = 1
        RangeIndices(2) = z_sRowToIndex(CellsArray(1))
        RangeIndices(3) = 16383
    Else
        Stop
    End If
End If
z_RangeToIndices = RangeIndices
End Function

Function z_sCellToIndex(ByRef CellIndexStr As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim CellIndexArr() As String 'Splited R1C1 Format
    Dim ColIndex As Integer
    Dim RowIndex As Integer
    Dim CellIndices(0 To 1) As Long
    'find column index
    CellIndexArr = Split(CellIndexStr, "C")
    ColIndex = CInt(CellIndexArr(1))
    CellIndexArr = Split(CellIndexArr(0), "R")
    RowIndex = CInt(CellIndexArr(1))
    CellIndices(0) = RowIndex
    CellIndices(1) = ColIndex
    'Output
    z_sCellToIndex = CellIndices
End Function

Function z_sColumnToIndex(ByRef ColIndexStrLeft As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim ColArray() As String
    ColArray = Split(ColIndexStrLeft, "C")
    z_sColumnToIndex = ColArray(1)
End Function

Function z_sRowToIndex(ByRef RowIndexStrUp As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim RowArray() As String
    RowArray = Split(RowIndexStrUp, "R")
    z_sRowToIndex = RowArray(1)
End Function
'*****Get Range Indices

```

## **File: Innovation Workbook Slide.pptm**

```

Sub m_ReNameAllShapesOfActiveSlide()
    Dim ap As presentation

```

```

Dim sl As Slide
Set ap = ActivePresentation
Dim WhichSlide As Integer
WhichSlide = InputBox("Which Slide? - Enter the slide number", , , 13500, 12500)
Set sl = ap.Slides(WhichSlide)
sl.Select
sl.Shapes.SelectAll
Dim sr As ShapeRange
Set sr = ap.Windows(1).Selection.ShapeRange
Dim s As Shape
For Each s In sr
    s.Select
    Name$ = s.Name

    Name$ = InputBox$("Give this shape a name. To stop enter: x", "Shape Name", Name$, 13500,
12500)

    If Name$ = "x" Or Name$ = "X" Then
        Exit Sub
    End If

    If Name$ <> "" Then
        ActiveWindow.Selection.ShapeRange(1).Name = Name$
    End If

Next

End Sub

Sub m_imbedExcelTable()
'do that via ribbon: insert object as icon (not linked)
End Sub

Sub m_AddTextFromExcelSheet()
Dim PPPres As PowerPoint.presentation
Dim PPShp As PowerPoint.Shape
Dim PPSld As PowerPoint.Slide

Dim SlideNum As Integer
Dim ShapeName As String
Dim ShapeNum As Integer

'set presentation
Set PPPres = ActivePresentation

'set embedded Excel data source (in: PPSld; out: PPShp and PPWb)
Dim PPSld_Wb As PowerPoint.Slide
Dim PPShp_Wb As PowerPoint.Shape
Dim PPWb As Excel.Workbook
Set PPSld_Wb = PPPres.Slides(1)
Call z_PPSld_PPShpAsOleFormatObject_Find(PPShp_Wb, PPWb, PPSld_Wb)
Dim WSh_1 As Worksheet

```

```

Dim WSh_2 As Worksheet
Dim WSh_3 As Worksheet
Dim WSh_4 As Worksheet
Set WSh_1 = PPWb.Worksheets("PIData")
Set WSh_2 = PPWb.Worksheets("MilestonesData")
Set WSh_3 = PPWb.Worksheets("StageMapping")
Set WSh_4 = PPWb.Worksheets("InvestmentSegMapping")
Set WSh_5 = PPWb.Worksheets("TechnologyColor")
'Stop

'loop through the slides
Dim FirstSlideToWriteInto As Long
FirstSlideToWriteInto = 6
For SlideNum = FirstSlideToWriteInto To PPPres.Slides.Count
    If SlideNum = 86 Then
        'Stop
    End If
    Debug.Print SlideNum
    *****

    'Set the Row in function of the slidenum
    Row = SlideNum - 4

    'Tb1: Confetti Color
    ShapeName = "Tb1"
    Technology = WSh_1.Cells(Row, 4)
    If Mid(Technology, 3) = "Genetics" Then
        PPPres.Slides(SlideNum).Shapes(ShapeName).Fill.ForeColor.RGB = RGB(170, 180, 0)
    ElseIf Mid(Technology, 3) = "New & Integrated Technology" Then
        PPPres.Slides(SlideNum).Shapes(ShapeName).Fill.ForeColor.RGB = RGB(235, 130, 0)
    ElseIf Mid(Technology, 3) = "Chemicals" Then
        PPPres.Slides(SlideNum).Shapes(ShapeName).Fill.ForeColor.RGB = RGB(0, 160, 190)
    ElseIf Mid(Technology, 3) = "Global Functions & Capabilities" Then
        PPPres.Slides(SlideNum).Shapes(ShapeName).Fill.ForeColor.RGB = RGB(112, 48, 160)
    Else
        Stop
    End If

    'Tb1a: Technology
    ShapeName = "Tb1a"
    Technology = WSh_1.Cells(Row, 4)
    If Technology = "2.New & Integrated Technology" Then
        Technology_ = "2.New & Int Tech"
    ElseIf Technology = "4.Global Functions & Capabilities" Then
        Technology_ = "2.GI Func & Cap"
    Else
        Technology_ = Technology
    End If
    PPPres.Slides(SlideNum).Shapes(ShapeName). _
        TextFrame.TextRange.Text = Mid(Technology_, 3)
    'Tb1b: PI SubType, Investment Segment
    ShapeName = "Tb1b"
    PI_SubType = WSh_1.Cells(Row, 5)
    If PI_SubType = "AI New" Then

```

```

    PI_SubType = "New AI"
Elseif PI_SubType = "Capability & Technology Development" Then
    PI_SubType = "Capability & Tech Dev"
Elseif PI_SubType = "Formulation Extension" Then
    PI_SubType = "Form Ext"
Elseif PI_SubType = "Formulation New" Then
    PI_SubType = "Form New"
Elseif PI_SubType = "GM Trait Stack Extension" Then
    PI_SubType = "GM Trait Stack Ext"
Elseif PI_SubType = "Idea Evaluation" Then
    PI_SubType = "Idea Eval"
Elseif PI_SubType = "Label Extension" Then
    PI_SubType = "Label Ext"
Elseif PI_SubType = "Non-Product Customer Offer" Then
    PI_SubType = "Non-Product Cust Offer"
Else
    'only the long ones to truncate are listed above.
End If
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    TextFrame.TextRange.Text = PI_SubType
'Tb1c: Investment Segment
ShapeName = "Tb1c"
InvestmentSegment = WSh_1.Cells(Row, 6)
If Mid(InvestmentSegment, 5) = "Breeding" Then
    InvestmentSegment = "Breeding"
Elseif Mid(InvestmentSegment, 5) = "Integrated Solutions" Then
    InvestmentSegment = "Int Soln"
Elseif Mid(InvestmentSegment, 5) = "Genetic Modification Trait" Then
    InvestmentSegment = "GM Trait"
Elseif Mid(InvestmentSegment, 5) = "Native Traits" Then
    InvestmentSegment = "NT"
Elseif Mid(InvestmentSegment, 5) = "Herbicides" Then
    InvestmentSegment = "H"
Elseif Mid(InvestmentSegment, 5) = "Fungicides" Then
    InvestmentSegment = "F"
Elseif Mid(InvestmentSegment, 5) = "Crop Enhancement" Then
    InvestmentSegment = "CE"
Elseif Mid(InvestmentSegment, 5) = "Seed Care" Then
    InvestmentSegment = "SC"
Elseif Mid(InvestmentSegment, 5) = "Insecticides" Then
    InvestmentSegment = "I"
Elseif Mid(InvestmentSegment, 5) = "Adjacent Technology" Then
    InvestmentSegment = "Adj Tech"
Elseif Mid(InvestmentSegment, 5) = "Bio Controls" Then
    InvestmentSegment = "Bio Controls"
Else
    'Stop
End If
If Mid(Technology, 3) = "Genetics" Then
    PPPres.Slides(SlideNum).Shapes(ShapeName).TextFrame.TextRange.Text = ""
Elseif Mid(Technology, 3) = "New & Integrated Technology" Then
    PPPres.Slides(SlideNum).Shapes(ShapeName). _
        TextFrame.TextRange.Text = InvestmentSegment

```

```

Elseif Mid(Technology, 3) = "Chemicals" Then
    PPPres.Slides(SlideNum).Shapes(ShapeName). _
        TextFrame.TextRange.Text = InvestmentSegment
Elseif Mid(Technology, 3) = "Global Functions & Capabilities" Then
    PPPres.Slides(SlideNum).Shapes(ShapeName).TextFrame.TextRange.Text = ""
Else
    Stop
End If
'Tb2: PI Title
ShapeName = "Tb2"
PITitle = WSh_1.Cells(Row, 2)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    TextFrame.TextRange.Text = PITitle 'Mid(PITitle, 1, 48)
'Tb3: PI Identifier
ShapeName = "Tb3"
PIId = WSh_1.Cells(Row, 1)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    TextFrame.TextRange.Text = PIId
'Tb4: PI Manager
ShapeName = "Tb4"
PIManager = WSh_1.Cells(Row, 3)
PIManager_lastName = ""
PIManager_firstName = ""
PIManager_arr = Split(PIManager, " ")
PIManager_lastName = PIManager_arr(LBound(PIManager_arr))
For iter = LBound(PIManager_arr) + 1 To UBound(PIManager_arr)
    PIManager_firstName = PIManager_firstName & " " & PIManager_arr(iter)
Next
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    TextFrame.TextRange.Text = Mid(PIManager_firstName, 2) & " " & PIManager_lastName
'
'
ShapeName = "Test2"
PPPres.Slides(SlideNum).Shapes(ShapeName).TextFrame.TextRange.Text = "Test: " & PIManager
'Tb5: PI Planned Start
ShapeName = "Tb5"
PIPlannedStart = WSh_1.Cells(Row, 7)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    TextFrame.TextRange.Text = Year(PIPlannedStart)
'Tb6: BC First Year of Sales
ShapeName = "Tb6"
PILaunchYear = WSh_1.Cells(Row, 8)
If PILaunchYear <> Empty Then
    PPPres.Slides(SlideNum).Shapes(ShapeName). _
        TextFrame.TextRange.Text = Year(PILaunchYear)
End If
'Tb7: PI Scope
ShapeName = "Tb7"
PIScope = WSh_1.Cells(Row, 9)
'remove Chr(10)
Find = Chr(10)
Replacement = Chr(32)
PIScope = Replace(PIScope, Find, Replacement)
'remove Chr(13)
Find = Chr(13)

```

```

Replacement = Chr(32)
PIScope = Replace(PIScope, Find, Replacement)
'remove Chr(9)
Find = Chr(9)
Replacement = Chr(32)
PIScope = Replace(PIScope, Find, Replacement)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    TextFrame.TextRange.Text = Mid(PIScope, 1, 300)
'Tb8: PI Business Rational and Benefits
ShapeName = "Tb8"
PIBusinessRational = WSh_1.Cells(Row, 10)
'remove Chr(10)
Find = Chr(10)
Replacement = Chr(32)
PIBusinessRational = Replace(PIBusinessRational, Find, Replacement)
'remove Chr(13)
Find = Chr(13)
Replacement = Chr(32)
PIBusinessRational = Replace(PIBusinessRational, Find, Replacement)
'remove Chr(9)
Find = Chr(9)
Replacement = Chr(32)
PIBusinessRational = Replace(PIBusinessRational, Find, Replacement)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    TextFrame.TextRange.Text = Mid(PIBusinessRational, 1, 360)
'Tb9: PI Main Risks
ShapeName = "Tb9"
PIRisks = WSh_1.Cells(Row, 11)
'remove Chr(10)
Find = Chr(10)
Replacement = Chr(32)
PIRisks = Replace(PIRisks, Find, Replacement)
'remove Chr(13)
Find = Chr(13)
Replacement = Chr(32)
PIRisks = Replace(PIRisks, Find, Replacement)
'remove Chr(9)
Find = Chr(9)
Replacement = Chr(32)
PIRisks = Replace(PIRisks, Find, Replacement)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    TextFrame.TextRange.Text = Mid(PIRisks, 1, 360)
*****
'T1-2.2: PI Lead Strategic Crop
ShapeName = "T1"
PILeadStrategicCrop = WSh_1.Cells(Row, 13)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    Table.Cell(2, 2).Shape.TextFrame.TextRange.Text = PILeadStrategicCrop
'T1-3.2: PI Customer Need
ShapeName = "T1"
PICustomerNeed = WSh_1.Cells(Row, 14)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    Table.Cell(3, 2).Shape.TextFrame.TextRange.Text = PICustomerNeed

```

```

'T1-4.2: PI Responsibility
ShapeName = "T1"
PIResponsibility = WSh_1.Cells(Row, 15)
PIResponsibility_arr = Split(PIResponsibility, " ")
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    Table.Cell(4, 2).Shape.TextFrame.TextRange.Text = PIResponsibility_arr(0)
'T1-5.2: PI Ext Competitor Dyn
ShapeName = "T1"
PIExtCompetitorDyn = WSh_1.Cells(Row, 11)
'remove Chr(10)
Find = Chr(10)
Replacement = Chr(32)
PIExtCompetitorDyn = Replace(PIExtCompetitorDyn, Find, Replacement)
'remove Chr(13)
Find = Chr(13)
Replacement = Chr(32)
PIExtCompetitorDyn = Replace(PIExtCompetitorDyn, Find, Replacement)
'remove Chr(9)
Find = Chr(9)
Replacement = Chr(32)
PIExtCompetitorDyn = Replace(PIExtCompetitorDyn, Find, Replacement)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    Table.Cell(5, 2).Shape.TextFrame.TextRange.Text = Mid(PIExtCompetitorDyn, 1, 50)

*****

'T2-2.2: Incremental Peak Sales
ShapeName = "T2"
PIPeakSales = WSh_1.Cells(Row, 16)
If PIPeakSales = Empty Then
    PPPres.Slides(SlideNum).Shapes(ShapeName). _
        Table.Cell(2, 2).Shape.TextFrame.TextRange.Text = "na"
Else
    PPPres.Slides(SlideNum).Shapes(ShapeName). _
        Table.Cell(2, 2).Shape.TextFrame.TextRange.Text = Format(PIPeakSales / 1000000,
"$#,##0.0") & "m"
End If
'T2-3.2: EAC Full Costs 2013
ShapeName = "T2"
EACFullCost2013 = WSh_1.Cells(Row, 17)
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    Table.Cell(3, 2).Shape.TextFrame.TextRange.Text = Format(EACFullCost2013 / 1000000,
"$#,##0.0") & "m"
'T2-4.2: BC NPV
ShapeName = "T2"
BCNPV = WSh_1.Cells(Row, 18)
If BCNPV = Empty Then
    PPPres.Slides(SlideNum).Shapes(ShapeName). _
        Table.Cell(4, 2).Shape.TextFrame.TextRange.Text = "na"
Else
    PPPres.Slides(SlideNum).Shapes(ShapeName). _
        Table.Cell(4, 2).Shape.TextFrame.TextRange.Text = Format(BCNPV / 1000000, "$#,##0.0") &
"m"
End If

```

```

*****

'Loop through Milestones
PiMs = 2
RowSize = z_RowSize(1, WSh_2.Name, PPWb)
If Not WSh_2.Range(WSh_2.Cells(2, 1), WSh_2.Cells(RowSize, 1)).Find(PIId, , , xlWhole) Is
Nothing Then
    For RowPiMs = 2 To RowSize
        If WSh_2.Cells(RowPiMs, 1) = PIId Then
            'T3-PiMs.1: MS1 Activity Title
            ShapeName = "T3"
            ActivityTitle = WSh_2.Cells(RowPiMs, 3)
            On Error GoTo AddTableRow:
            PPPres.Slides(SlideNum).Shapes(ShapeName). _
                Table.Cell(PiMs, 1).Shape.TextFrame.TextRange.Text = ActivityTitle
            On Error GoTo 0
            'T3-PiMs.2: MS1 Activity Date
            ShapeName = "T3"
            ActivityDate = WSh_2.Cells(RowPiMs, 4)
            mm = Month(ActivityDate)
            If mm < 10 Then
                mm = "0" & mm
            End If
            yyyy = Year(ActivityDate)
            PPPres.Slides(SlideNum).Shapes(ShapeName). _
                Table.Cell(PiMs, 2).Shape.TextFrame.TextRange.Text = mm & " / " & yyyy
            'Format(ActivityDate, "mm - yyyy")
            'iterate
            PiMs = PiMs + 1
        End If
    Next
End If
*****

'Stage
PIStage = WSh_1.Cells(Row, 19)
ShapeName = "R2"
PPPres.Slides(SlideNum).Shapes(ShapeName).LockAspectRatio = msoFalse
PPPres.Slides(SlideNum).Shapes(ShapeName).Width = 41
If PIStage = "Seeds-Discovery" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(0 + 1 / 8, msoFalse)
ElseIf PIStage = "Seeds-Proof of Concept" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(0 + 1 / 4, msoFalse)
ElseIf PIStage = "Seeds-1" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(0 + 1 / 2, msoFalse)
ElseIf PIStage = "Seeds-2" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(0 + 3 / 4, msoFalse)
ElseIf PIStage = "Seeds-3" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(2 + 1 / 2, msoFalse)
ElseIf PIStage = "Seeds-4" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(3 + 1 / 4, msoFalse)
ElseIf PIStage = "Seeds-5" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(3 + 1 / 2, msoFalse)
ElseIf PIStage = "Seeds-6" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(4 + 1 / 2, msoFalse)

```



```

Elseif PISStage = "Seeds-7" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(5 + 1 / 8, msoFalse)
Elseif PISStage = "Seeds-8" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(5 + 1 / 4, msoFalse)
Elseif PISStage = "Seeds-9" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(5 + 1 / 2, msoFalse)
Elseif PISStage = "Seeds-10" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(5 + 3 / 4, msoFalse)
Elseif PISStage = "Seeds-11" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(5 + 3 / 4, msoFalse)
Elseif PISStage = "Seeds-Discontinue" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(6, msoFalse)
Elseif PISStage = "Seeds-Early Development" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(2 + 1 / 2, msoFalse)
Elseif PISStage = "Seeds-Late Development" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(3 + 1 / 2, msoFalse)
Elseif PISStage = "Seeds-Pre-Commercial" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(4 + 1 / 2, msoFalse)
Elseif PISStage = "Seeds-Commercial" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(5 + 1 / 2, msoFalse)
Elseif PISStage = "CP-1-Research" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(1 + 1 / 2, msoFalse)
Elseif PISStage = "CP-2-Evaluation" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(2 + 1 / 2, msoFalse)
Elseif PISStage = "CP-3-Development" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(3 + 1 / 2, msoFalse)
Elseif PISStage = "CP-4-Life Cycle Mgmt" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(4, msoFalse)
Elseif PISStage = "CP-A-Feasibility" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(3 + 1 / 8, msoFalse)
Elseif PISStage = "CP-B-Evaluation" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(3 + 1 / 4, msoFalse)
Elseif PISStage = "CP-C-Development" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(3 + 1 / 2, msoFalse)
Elseif PISStage = "CP-D-Sales" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(4, msoFalse)
Elseif PISStage = "CP-Not Applicable" Then
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(1 / 100, msoFalse)
Else
    'Stop
    Call PPPres.Slides(SlideNum).Shapes(ShapeName).ScaleWidth(1 / 100, msoFalse)
End If
' 'for debugging purposes
' ShapeName = "Test1"
' PPPres.Slides(SlideNum).Shapes(ShapeName).TextFrame.TextRange.Text = "Test: " & PISStage

Next
Exit Sub
AddTableRow:
'Stop
PPPres.Slides(SlideNum).Shapes(ShapeName). _
    Table.Rows.Add
Resume

```

End Sub

```
Function z_PPSld_PPSHpAsOleFormatObject_Find( _  
    Optional ByRef PPSHp_out As PowerPoint.Shape, Optional ByRef Wb_out As Excel.Workbook,  
    -  
    Optional ByRef PPSld As PowerPoint.Slide, Optional ByRef PPPres As PowerPoint.presentation,  
    -  
    Optional Index_ As Integer, Optional Slideld_ As String)  
    'Call z_PPSld_PPSHpAsOleFormatObject_Find(PPShp, PPWb, PPSld)  
    'Call z_PPSld_PPSHpAsOleFormatObject_Find(PPShp, PPWb, ,PPPres, 1)  
    'find slide  
    If PPSld Is Nothing Then  
        If Index_ <> Empty Then  
            Set PPSld = PPPres.Slides(Index_)  
        ElseIf Slideld_ <> Empty Then  
            For iter = 2 To PPPres.Slides.Count  
                If PPPres.Slides(iter).SlideID = Slideld_ Then  
                    Set PPSld = PPPres.Slides(iter)  
                    Exit For  
                End If  
            Next  
        Else  
            Stop  
        End If  
    End If  
    'find shape  
    For Each PPSHp_iter In PPSld.Shapes  
        If PPSHp_iter.Type = msoEmbeddedOLEObject Then  
            Set PPSHp_out = PPSHp_iter  
            Exit For  
        End If  
    Next  
    'find Wb  
    Set Wb_out = PPSHp_out.OLEFormat.Object  
End Function
```

```
Function z_RowSize(SearchCol As Long, Sh As String, ByRef wb As Workbook) As Long  
    'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)  
    'Version without selection of worksheets!  
    z_RowSize = If(IsEmpty(wb.Sheets(Sh).Cells(1048576, SearchCol)), wb.Sheets(Sh).Cells(1048576,  
    SearchCol).End(xlUp).Row, 1048576)  
End Function
```

## **File: TimeCardActualsByWeek\_V2**

```
Sub SmC_CleanRBS_Extract()  
    'A user of this macro needs to provide an extract of the SmartChoice RBS which is limited to 4 levels.  
    'Otherwise the macro will stop and ask the user to first generate such an extract.
```

Dim IntRBSValue As String, IntRBSValue2 As String

'Column TimeCardManager must exist and must be blank for all rows. Level 1-4 is reserved for RBS

'Level 5-.. can be (mis-)used for other tasks

```
If Not WorksheetFunction.CountBlank(Range(Rows("1:1").Find(What:="TimeCard Manager",  
LookAt:=xlWhole).Offset(1, 0), Rows("1:1").Find(What:="TimeCard Manager",  
LookAt:=xlWhole).Offset(1000, 0))) = Range(Rows("1:1").Find(What:="TimeCard Manager",  
LookAt:=xlWhole).Offset(1, 0), Rows("1:1").Find(What:="TimeCard Manager",  
LookAt:=xlWhole).Offset(1000, 0)).Count Then
```

```
    MsgBox ("Please generate first a SmC RBS extract which is limited to level 4 before running this  
macro")
```

```
    Exit Sub
```

```
End If
```

'MsgBox to activate the right sheet, &vbCrLf &

```
If 1 Then
```

```
    Response = MsgBox( _  
        "Please select the Excel sheet you want the macro to be applied to!" & Chr(13) & _  
        " " & Chr(13) & _  
        "Click Yes: if the right sheet is already selected" & Chr(13) & _  
        "      Otherwise" & Chr(13) & _  
        "Click No: and select the Excel sheet by clicking on its tab at the" & Chr(13) & _  
        "      bottom and then restart the macro", _  
        vbYesNo)
```

```
End If
```

```
If Response = vbYes Then
```

'The active sheet will be renamed and a new sheet RBS Table will be created

```
ActiveSheet.Name = "RBS Extract"
```

```
Sheets("RBS Extract").Copy Before:=Sheets("RBS Extract")
```

```
ActiveSheet.Name = "RBS Table"
```

```
' Rows("2:2").Select
```

```
' ActiveWindow.FreezePanels = True
```

'Generally "unmerge" merged cells

```
Cells.Select
```

```
Selection.MergeCells = False
```

'Remove the ALL level

```
If Cells(2, 7).Value = "ALL" Then
```

```
    Cells(2, 7).EntireRow.Delete
```

```
End If
```

'Remove not used columns

```
'Columns("A:B").EntireColumn.Delete
```

```
Columns("E:E").EntireColumn.Delete
```

```
'Columns("A:A").EntireColumn.Delete
```

'Add headers for the columns

```
Range("A:H").ColumnWidth = 20
```

```
Cells(1, 1).Value = "RBS Level 1 Name"
```

```
Cells(1, 3).Value = "RBS Level 2 Name"
Cells(1, 5).Value = "RBS Level 3 Name"
Cells(1, 2).Value = "RBS Level 1 Description"
Cells(1, 4).Value = "RBS Level 2 Description"
Cells(1, 6).Value = "RBS Level 3 Description"
```

'Assign RBS level 1, 2 and 3 values to the cells on the left of each entry

'RBS level 1

Cells(2, 5).Select

Do While Not ActiveCell.Value = ""

If Not ActiveCell(1, -2).Borders(xlEdgeRight).LineStyle = xlContinuous Then

IntRBSValue = ActiveCell.Value

IntRBSValue2 = ActiveCell(1, 2).Value

ActiveCell.EntireRow.Delete

Do While ActiveCell(1, -2).Borders(xlEdgeRight).LineStyle = xlContinuous

ActiveCell(1, -3).Value = IntRBSValue

ActiveCell(1, -2).Value = IntRBSValue2

ActiveCell(2, 1).Select

Loop

End If

Loop

'RBS level 2

Cells(2, 5).Select

Do While Not ActiveCell.Value = ""

If Not ActiveCell(1, -1).Borders(xlEdgeRight).LineStyle = xlContinuous Then

IntRBSValue = ActiveCell.Value

IntRBSValue2 = ActiveCell(1, 2).Value

ActiveCell.EntireRow.Delete

Do While ActiveCell(1, -1).Borders(xlEdgeRight).LineStyle = xlContinuous

ActiveCell(1, -1).Value = IntRBSValue

ActiveCell(1, 0).Value = IntRBSValue2

ActiveCell(2, 1).Select

Loop

End If

Loop

'Remove outdated no more used RBS entries prefixed with "XXX\_"

Cells(2, 5).Select

Do While Not ActiveCell.Value = ""

If Left(ActiveCell, 4) = "XXX\_" Then

ActiveCell.EntireRow.Delete

Else: ActiveCell(2, 1).Select

End If

Loop

'Remove not used columns

Columns("G:G").EntireColumn.Delete

Columns("H:P").EntireColumn.Delete

Cells(1, 1).Select

Else

Exit Sub

End If

End Sub

Sub SmC\_TimeCardActuals()

'Macro generated by Roland.Benz@Syngenta.com and franz.schuermann@syngenta.com (PMEC,  
Project Management Excellence)

'Date: 26.9.2011

'Macro for Lee Hubbard

Dim Start As Date: Dim Duration As Date

Start = Now()

'MsgBox to activate the right sheet, &vbCrLf &

If 1 Then

Response = MsgBox( \_

"Please select the Excel sheet you want the macro to be applied to!" & Chr(13) & \_

" " & Chr(13) & \_

"Click Yes: if the right sheet is already selected" & Chr(13) & \_

" Otherwise" & Chr(13) & \_

"Click No: and select the Excel sheet by clicking on its tab at the" & Chr(13) & \_

" bottom and then restart the macro", \_

vbYesNo)

End If

'Execute the tasks

'Copy the input file ActiveSheet.Name in a new file Sh\_Source and make changes

'Make new sheets Sh\_Pivot and make the pivots

If Response = vbYes Then

'Make a copy of the input sheet, change the sheet tab color and then make some some changes

Dim Sh As String

Sh = ActiveSheet.Name ' msgbox asks to make the right sheet active

Dim Sh\_Source As String

Sh\_Source = "ActualsByWeek"

Dim Sh\_Source2 As String

Sh\_Source2 = "RBS Table"

If 1 Then

Call z\_ShNewFlatValueCopy(Sh, Sh\_Source, "End")

'change the colour of the sheet name on the tab

With ActiveWorkbook.Sheets(Sh\_Source).Tab

.Color = RGB(0, 32, 90)

.TintAndShade = 0

End With

End If

If 1 Then

'Add a column for person site information

Sheets(Sh\_Source).Select

Rows("1:1").Find(What:="EMPLOYEE\_NAME", LookAt:=xlWhole).Select

ActiveCell.Offset(0, 1).EntireColumn.Insert

ActiveCell(1, 2).Value = "SITE"

'Fill the new column with person site info

Range(ActiveCell(2, 1), ActiveCell(2, 1).End(xlDown)).Select

For Each rSite In Selection.Cells

```

        rSite.Offset(0, 1).Value = Right(rSite.Value, 4)
    Next
End If
If 1 Then
    'Add a column for a concatenated string of PI ID and Task ID
    Sheets(Sh_Source).Select
    Rows("1:1").Find(What:="STAFF_DAYS", LookAt:=xlWhole).Select
    ActiveCell.Offset(0, 1).EntireColumn.Insert
    ActiveCell(1, 2).Value = "IDENTIFICATION"
    Range(ActiveCell(2, 2), ActiveCell(2, 1).End(xlDown).Offset(0, 1)).Select
    For Each rSite In Selection.Cells
        rSite.Value = rSite.Offset(0, -16).Value & "-" & rSite.Offset(0, -12).Value
    Next
    Range("A1").Select
End If
If 1 Then
    'Add a column for Budget Group and map the Budget group values from Sh_Source2 to
    Sh_Source
    'SmcSigma: SmCExtract: PMEC Term: Lee's term:
    'SmC Level 2->RBS Level 1->Resource Group->Budget Group
    'SmC Level 3->RBS Level 2->Resource
    'SmC Level 4->RBS Level 3->Resource Role -> Budget Center
    Sheets(Sh_Source).Select
    Rows("1:1").Find(What:="TIMECARD_COMMENT", LookAt:=xlWhole).Select
    ActiveCell.Offset(0, 1).EntireColumn.Insert
    ActiveCell(1, 2).Value = "BUDGET_GROUP"
    Call z_ShMapColumns(Sh_Source2, "RBS Level 3 Description", Array("RBS Level 1 Description"),
    Sh_Source, "BUDGET_CENTER", Array("BUDGET_GROUP"), "Log_1")
End If

'Make new sheets and the pivots from Sh_Source
Dim Sh_Pivot As String
Dim RowU, ColL, RowD, ColR As Long
Dim Piv_RowD As Long
Dim Source_Rng As Range
Dim Piv_sULCell As String
Dim Piv_ULCell As Range
Dim Piv_F_R_C_V(0 To 3) As Variant
Dim PivName(0 To 2) As String

'Make a new sheet and add a pivot
'Name of the sheet
Sh_Pivot = "TimeToDateByTask"
If 1 Then
    Call z_ShNew(Sh_Pivot, "End")
    'change the colour of the sheet name on the tab
    With ActiveWorkbook.Sheets(Sh_Pivot).Tab
        .Color = RGB(255, 255, 0)
        .TintAndShade = 0
    End With
End If
If 1 Then
    'Parameters for the z_GeneratePivotTable function and its call

```

```

'Name of the pivot sheet
    'Sh_Pivot = "Time-to-Date-By-Task-RB"
'Name of the source sheet
    'Sh_Source = "ActualsByWeek-RB"
'Determine the range of Sh_Source and create a range object
RowU = 1: ColL = 1
RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'Create a range object for the upper left corner of the pivot
Piv_sULCell = "B9"
Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
'Determine the pivot fields
Piv_F_R_C_V(0) = Array(13, 9, 11, 7, 12, 10) 'filter
Piv_F_R_C_V(1) = Array(17, 5) 'row labels
Piv_F_R_C_V(2) = Array() 'column labels
Piv_F_R_C_V(3) = Array(16) 'values
'Call the function that creates the pivot
PivName(0) = z_GeneratePivotTable(Piv_ULCell, Source_Rng, Sh_Source, Sh_Pivot, Piv_F_R_C_V)
End If
If 1 Then
    'Pivot PivName(0)
    'set the filters
    Dim CurrentYear As String
    CurrentYear = Year(Date)
    ActiveSheet.PivotTables(PivName(0)).PivotFields("RESOURCE_YEAR").ClearAllFilters
    ActiveSheet.PivotTables(PivName(0)).PivotFields("RESOURCE_YEAR").CurrentPage = CurrentYear
    'change the column with of col A
    Columns("A:A").ColumnWidth = 5
    'insert formula into the Sh_Pivot
    Sheets(Sh_Pivot).Select
    Range("D7").Select
    Selection.Formula = "=GETPIVOTDATA("""STAFF_DAYS""",$B$9)"
    'format columns in Sh_Pivot
    Range("D2:D7").Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent3
        .TintAndShade = 0.599993896298105
        .PatternTintAndShade = 0
    End With
    Columns("A:A").ColumnWidth = 3
    Columns("D:D").Select
    Selection.NumberFormat = "#,##0.0"
    ActiveWorkbook.Save
    Range("B9").Select
End If

'Make a new sheet and add two pivots
'Name of the sheet
Sh_Pivot = "DataQualityLocation"
If 1 Then
    Call z_ShNew(Sh_Pivot, "End")

```

```

'change the colour of the sheet name on the tab
With ActiveWorkbook.Sheets(Sh_Pivot).Tab
.Color = RGB(243, 130, 37)
.TintAndShade = 0
End With
End If
If 1 Then
'Parameters for the z_GeneratePivotTable function and its call
'Name of the pivot sheet
'Sh_Pivot = "DataQualityLocation-RB"
'Name of the source sheet
'Sh_Source = "ActualsByWeek-RB"
'Determine the range of Sh_Source and create a range object
RowU = 1: ColL = 1
RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'Create a range object for the upper left corner of the pivot
Piv_sULCell = "B7"
Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
'Determine the pivot fields
Piv_F_R_C_V(0) = Array(13, 9) 'filter (enter numbers in proper order)
Piv_F_R_C_V(1) = Array(12, 19) 'row labels (enter numbers in reverse order)
Piv_F_R_C_V(2) = Array(7) 'column labels
Piv_F_R_C_V(3) = Array(16) 'values
'Call the function that creates the pivot
PivName(1) = z_GeneratePivotTable(Piv_ULCell, Source_Rng, Sh_Source, Sh_Pivot, Piv_F_R_C_V)
End If
If 1 Then
'Parameters for the z_GeneratePivotTable function and its call
'Name of the sheet
'Sh_Pivot = "DataQualityLocation-RB"
'Determine the range of Sh_Source and create a range object
RowU = 1: ColL = 1
RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'Create a range object for the upper left corner of the pivot
Piv_RowD = z_RowSize(2, Sh_Pivot)
Piv_sULCell = "B" & Piv_RowD + 8 'place the second pivot under the first pivot
Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
'Determine the pivot fields
Piv_F_R_C_V(0) = Array(13, 9) 'filter (enter numbers in proper order)
Piv_F_R_C_V(1) = Array(17, 11, 19) 'row labels (enter numbers in reverse order)
Piv_F_R_C_V(2) = Array(7) 'column labels
Piv_F_R_C_V(3) = Array(16) 'values
'Call the function that creates the pivot
PivName(2) = z_GeneratePivotTable(Piv_ULCell, Source_Rng, Sh_Source, Sh_Pivot, Piv_F_R_C_V)
End If
'Add some formats and texts
If 1 Then
'Pivot PivName(1)
'set the filters
ActiveSheet.PivotTables(PivName(1)).PivotFields("RESOURCE_YEAR").ClearAllFilters
ActiveSheet.PivotTables(PivName(1)).PivotFields("RESOURCE_YEAR").CurrentPage = CurrentYear

```



```

ActiveSheet.PivotTables(PivName(1)).PivotFields("TASK_LOCATION").ClearAllFilters
ActiveSheet.PivotTables(PivName(1)).PivotFields("TASK_LOCATION").CurrentPage = "(blank)"
'change the column with of col A
Columns("A:A").ColumnWidth = 5
'add some text
Range("B2").Select
ActiveCell.Value = "Number of Tasks with recorded staff-days, Task Location =0"
Range("E2").Select
ActiveCell.Value = "241"
'alignement
Range("B2:D2").Select
With Selection
    .HorizontalAlignment = xlLeft
    .VerticalAlignment = xlBottom
    .WrapText = False
    .Orientation = 0
    .AddIndent = False
    .IndentLevel = 0
    .ShrinkToFit = False
    .ReadingOrder = xlContext
    .MergeCells = False
End With
'merge cells
Range("B2:D2").Select
Selection.Merge
'change the font
Range("B2:E2").Select
Selection.Font.Bold = True
'add borders
Range("B2:E2").Select
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThick
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThick
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThick
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .ColorIndex = 0

```

```

        .TintAndShade = 0
        .Weight = xlThick
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThick
    End With
    With Selection.Borders(xlInsideHorizontal)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThick
    End With
    'freeze panes
    Rows("9:9").Select
    ActiveWindow.FreezePanes = True
    'Pivot PivName(2)
    'copy/paste some text
    Range("B2:E2").Select
    Selection.Copy
    Range("B" & Piv_RowD + 3).Select
    ActiveSheet.Paste
    'add filters
    ActiveSheet.PivotTables(PivName(2)).PivotFields("RESOURCE_YEAR").ClearAllFilters
    ActiveSheet.PivotTables(PivName(2)).PivotFields("RESOURCE_YEAR").CurrentPage = CurrentYear
    ActiveSheet.PivotTables(PivName(2)).PivotFields("TASK_LOCATION").ClearAllFilters
    ActiveSheet.PivotTables(PivName(2)).PivotFields("TASK_LOCATION").CurrentPage = "(blank)"
    Range("B7").Select
End If
Else
    Exit Sub
End If

'save the workbook
'ActiveWorkbook.Save
Duration = Now() - Start
Debug.Print Duration
End Sub

Private Function z_ShMapColumns(Sh_from As String, ColName_Key_from As String, ByRef
ColNames_from As Variant, _
    Sh_to As String, ColName_Key_to As String, ByRef ColNames_to As Variant, _
    Optional Sh_log As String, Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Application.ScreenUpdating = False
'Start time measuring
Dim Start As Date: Dim Duration As Date
Start = Now()

'create a matrix with column names and indexes

```

```

MapMatrix = MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from,
ColNames_from, Sh_to, ColNames_to)

'Find the column index of the KeyName in "Sh_from"
Dim ColIndex_Key_from As Long
ColIndex_Key_from = z_GetColumnIndex(ColName_Key_from, 1, Sh_from)

'Find the column index of the KeyName in "Sh_to"
Dim ColIndex_Key_to As Long
ColIndex_Key_to = z_GetColumnIndex(ColName_Key_to, 1, Sh_to)

'Determine the row size in Sh_to
Sheets(Sh_to).Activate
RowSize_to = z_RowSize(ColIndex_Key_to, Sh_to)

'Select the range in the column Key_to
Sheets(Sh_to).Activate
Range(Cells(2, ColIndex_Key_to), Cells(RowSize_to, ColIndex_Key_to)).Select

'Iterate through the rows with "rcheck" = Pildentifier
Dim ilog As Long
ilog = 2
For Each ValueInCol_Key_to In Selection.Cells
    'if ValueInCol_Key_to is found in Sh_from then perform the mapping
    If Not Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to,
LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns with the help of the MapMatrix
        For j = LBound(MapMatrix) To UBound(MapMatrix) Step 1
            'read the indices
            ColIndex_from_j = MapMatrix(j, 1)
            ColIndex_to_j = MapMatrix(j, 3)
            'map
            ValueInCol_Key_to.Offset(0, (ColIndex_to_j) - ColIndex_Key_to).Value = _
                Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to, _
                LookAt:=xlWhole).Offset(0, (ColIndex_from_j) - ColIndex_Key_from)
        Next j
    Else
        'write not found ValueInCol_Key_to in Sh_from into Sh_log
        On Error Resume Next
        Sheets(Sh_log).Cells(ilog, 2) = ValueInCol_Key_to
        Sheets(Sh_log).Cells(ilog, 4) = " not found, map them from another source file Sh_from"
        ilog = ilog + 1
        On Error GoTo 0
    End If
Next

'In case the mapping has changed the row height
Sheets(Sh_to).Activate
Cells.Select
Selection.Rows.RowHeight = 15
Application.ScreenUpdating = True
'Write the durations into the logfile
On Error Resume Next

```

```

Duration = Now() - Start
Sheets(Sh_log).Cells(iLog + 2, 1) = "Duration" & CStr(Duration)
On Error GoTo 0
End Function

```

```

Private Function z_ChkColExistence(Sh As String, ByRef ColNames As Variant, Sh_log As String) As Boolean

```

```

    'The column existence check is assumed to find all column names at the beginning
    z_ChkColExistence = True
    'Determine the column indices of the ColNames array in Sh
    Dim ColName_i As String
    ReDim Matrix_ColNameColIndex(0 To UBound(ColNames), 0 To 1) As Variant
    'iterate through the array
    For i = LBound(ColNames_from) To UBound(ColNames) Step 1
        ColName_i = CStr(ColNames(i))
        Matrix_ColNameColIndex(i, 0) = ColName_i
        Matrix_ColNameColIndex(i, 1) = z_GetColumnIndex(ColName_i, 1, Sh)
        Debug.Print Matrix_ColNameColIndex(i, 0) & " " & Matrix_ColNameColIndex(i, 1)
        If Matrix_ColNameColIndex(i, 1) = 0 Then
            'write errors into the logfile
            Sheets(Sh_log).Cells(iLog, 2) = "ColName: "
            Sheets(Sh_log).Cells(iLog, 3) = Matrix_ColNameColIndex(i, 0)
            Sheets(Sh_log).Cells(iLog, 4) = "not found in " & Sh
            iLog = iLog + 1
            'The column existence check has detected an unfound column name
            z_ChkColExistence = False
        End If
    Next i
End Function

```

```

Private Function MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from As String, ByRef ColNames_from As Variant, _

```

```

    Sh_to As String, ByRef ColNames_to As Variant) As Variant
    'Determine the column indices in Sh and Sh_new,
    Dim ColName_from_i As String
    Dim ColName_to_i As String
    ReDim Matrix_ColName1Index1_ColName2Index2(0 To UBound(ColNames_from), 0 To 3) As Variant

    For i = LBound(ColNames_from) To UBound(ColNames_from) Step 1
        ColName_from_i = CStr(ColNames_from(i))
        Matrix_ColName1Index1_ColName2Index2(i, 0) = ColName_from_i
        Matrix_ColName1Index1_ColName2Index2(i, 1) = z_GetColumnIndex(ColName_from_i, 1, Sh_from)
        ColName_to_i = CStr(ColNames_to(i))
        Matrix_ColName1Index1_ColName2Index2(i, 2) = ColName_to_i
        Matrix_ColName1Index1_ColName2Index2(i, 3) = z_GetColumnIndex(ColName_to_i, 1, Sh_to)
        Debug.Print Matrix_ColName1Index1_ColName2Index2(i, 0) & " " & Matrix_ColName1Index1_ColName2Index2(i, 1) & " " & Matrix_ColName1Index1_ColName2Index2(i, 2) & " " & Matrix_ColName1Index1_ColName2Index2(i, 3)
    Next i
    MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex = Matrix_ColName1Index1_ColName2Index2
End Function

```

End Function

Private Function z\_ShNewFlatValueCopy(Sh As String, Sh\_new As String, Optional Where As String, Optional ByRef Sh\_Ref As Worksheet)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Name of the Sh to copy, Name of the new Sh\_new, where to place the new Sh\_new, before or after some Sh\_Ref, at the begin or the end

Call z\_ShAdd(Where, Sh\_Ref)

On Error Resume Next

ActiveSheet.Name = Sh\_new

If Err.Number <> 0 Then

Application.DisplayAlerts = False

ActiveSheet.Delete

Application.DisplayAlerts = True

Sheets(Sh\_new).Cells.ClearContents

End If

On Error GoTo 0

Sheets(Sh).Cells.Copy

Sheets(Sh\_new).Range("A1").Select

Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks \_  
:=False, Transpose:=False

Sheets(Sh\_new).Columns("A:GA").ColumnWidth = 20

End Function

Private Function z\_ShNew(Sh As String, Optional Where As String, Optional ByRef Sh\_Ref As Worksheet, Optional ByRef Wb As Workbook)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Name of the new Sh, where to place the new Sh, before or after some Sh\_Ref, at the begin or the end

On Error Resume Next

WorksheetExists = (Sheets(Sh).Name <> "")

On Error GoTo 0

If WorksheetExists = False Then

Call z\_ShAdd(Where, Sh\_Ref)

ActiveSheet.Name = Sh 'Worksheets.Add(Before:=Worksheets(1)).Name = Sh

End If

'Clear contents

Sheets(Sh).Activate

ActiveSheet.Cells.Select

Selection.ClearContents

'Format

Sheets(Sh).Columns.ColumnWidth = 20

Sheets(Sh).Rows.RowHeight = 15

End Function

Private Function z\_ShAdd(Optional Where As String, Optional ByRef Sh\_Ref As Worksheet)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: where to place the added Sh, before or after some Sh\_Ref, at the begin or the end

If Not Sh\_Ref Is Nothing Then

If Where = ("Before:=") Then

```

        Sheets.Add Sh_Ref
    ElseIf Where = ("After:=") Then
        Sheets.Add , Sh_Ref
    Else
        Sheets.Add Before:=Sheets(1)
    End If
Else
    If Where = ("End") Then
        Sheets.Add After:=Sheets(Sheets.Count)
    ElseIf Where = ("Begin") Then
        Sheets.Add Before:=Sheets(1)
    Else
        Sheets.Add Before:=Sheets(1)
    End If
End If
End Function

```

```

Private Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef MyWb As
Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet
Sheets(Sh).Activate
'Determine the row size
z_RowSize = If(IsEmpty(Cells(1048576, SearchCol)), Cells(1048576, SearchCol).End(xlUp).Row,
1048576)
End Function

```

```

Private Function z_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef MyWb As
Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Row, Output: column with the last entry in that row
'SearchCol datatype changed from integer
'Activate the Sheet
Sheets(Sh).Activate
'Determine the col size
z_ColSize = If(IsEmpty(Cells(SearchRow, 16384)), Cells(SearchRow, 16384).End(xlToLeft).Column,
16384)
End Function

```

```

Private Function z_GeneratePivotTable(Piv_ULCell As Range, Source_Rng As Range, _
        Sh_Source As String, Sh_Pivot As String, Piv_F_R_C_V As Variant) As String
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input parameters (arguments) for the z_GeneratePivotTable function and its call
'Name of the pivot sheet
'Sh_Pivot = "TimeToDateByTask"
'Name of the source sheet
'Sh_Source = "ActualsByWeek"
'Determine the range of Sh_Source and create a range object

```

```

'RowU = 1: ColL = 1
'RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
'Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'Create a range object for the upper left corner of the pivot
'Piv_sULCell = "B9"
'Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
'Determine the pivot fields
'Piv_F_R_C_V(0) = Array(13, 9, 11, 7, 12, 10) 'filter
'Piv_F_R_C_V(1) = Array(17, 5) 'row labels
'Piv_F_R_C_V(2) = Array() 'column labels
'Piv_F_R_C_V(3) = Array(16) 'values

'Creat the strings for the function ActiveWorkbook.PivotCaches.Create() further below
Dim sSource_Rng As String
sSource_Rng = Sh_Source & "!" & Source_Rng.Address(ReferenceStyle:=xlR1C1)
Dim sPivot_Rng As String
sPivot_Rng = Sh_Pivot & "!" & Piv_ULCell.Address(ReferenceStyle:=xlR1C1)

'Store the range.address information into an array
Dim RngAddress As Variant
RngAddress = z_RangeAddressAsArray(Source_Rng)

'Store the Column names into an array
ReDim PivChosenField(RngAddress(2) To RngAddress(4)) As String
For i = RngAddress(2) To RngAddress(4)
    PivChosenField(i) = Source_Rng.Cells(1, i)
Next i

'Store the column names of the ReportFilter, RowLabel, ColLabel and Value into arrays
ReDim PivReportFilter(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivRowLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivColLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivValue(RngAddress(2) - 1 To RngAddress(4) - 1) As String
For i = RngAddress(2) - 1 To RngAddress(4) - 1
    On Error Resume Next
    PivReportFilter(i) = PivChosenField(Piv_F_R_C_V(0)(i))
    On Error GoTo 0
    On Error Resume Next
    PivRowLabel(i) = PivChosenField(Piv_F_R_C_V(1)(i))
    On Error GoTo 0
    On Error Resume Next
    PivColLabel(i) = PivChosenField(Piv_F_R_C_V(2)(i))
    On Error GoTo 0
    On Error Resume Next
    PivValue(i) = PivChosenField(Piv_F_R_C_V(3)(i))
    On Error GoTo 0
Next i

'generate Pivot
Sheets(Sh_Pivot).Select
Piv_ULCell.Select
ActiveWorkbook.PivotCaches.Create( _
    SourceType:=xlDatabase, _

```

```

SourceData:=sSource_Rng, _
Version:=xlPivotTableVersion12).CreatePivotTable _
TableDestination:=sPivot_Rng, _
TableName:=Piv_Name, _
DefaultVersion:=xlPivotTableVersion12

```

```

'get Pivot table name
'if PivName = "PivotTable" is used more than once an iteger is added to the name
PivName = ActiveSheet.PivotTables(1).Name

```

```

For i = RngAddress(2) - 1 To RngAddress(4) - 1
'Define row labels
On Error Resume Next
With ActiveSheet.PivotTables(PivName).PivotFields(PivRowLabel(i))
.Orientation = xlRowField
.Position = 1
End With
On Error GoTo 0
'Define column labels
On Error Resume Next
With ActiveSheet.PivotTables(PivName).PivotFields(PivCollLabel(i))
.Orientation = xlColumnField
.Position = 1
End With
On Error GoTo 0
'Define values
On Error Resume Next
ActiveSheet.PivotTables(PivName).AddDataField ActiveSheet.PivotTables( _
PivName).PivotFields(PivValue(i)), "Sum of STAFF_DAYS", xlSum
On Error GoTo 0
'Define report filters
On Error Resume Next
With ActiveSheet.PivotTables(PivName).PivotFields(PivReportFilter(i))
.Orientation = xlPageField
.Position = 1
End With
On Error GoTo 0
Next i

```

```

'Change the layout
With ActiveSheet.PivotTables(PivName)
.InGridDropZones = True
.RowAxisLayout xlTabularRow
End With

```

```

'Define all colums from : Choose fields to add to report
For i = RngAddress(2) To RngAddress(4)
ActiveSheet.PivotTables(PivName).PivotFields(PivChosenField(i)).Subtotals = _
Array(False, False, False, False, False, False, False, False, False, False, False)
Next i

```

```

'output
z_GeneratePivotTable = PivName

```



End Function

```
Private Function z_RangeAddressAsArray(Rng As Range) As Variant
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Range(Cells(a,b),Cells(c,d)), Output: Array(a,b,c,d)
sRngAddress = Rng.Address(ReferenceStyle:=xlR1C1)
DltrLst = Array("$", "R", "C", ":")
RplLst = Array("@", "@", "@", "@")
For k = LBound(DltrLst) To UBound(DltrLst)
    If RplLst(k) = Empty Then
        l = RplLst(0)
    Else
        l = k
    End If
    sRngAddress = Replace(sRngAddress, DltrLst(k), RplLst(l))
Next k
sRngAddress = Replace(sRngAddress, RplLst(0) & RplLst(0), RplLst(l))
Dim RngAddress As Variant
RngAddress = Split(sRngAddress, RplLst(0))
For i = 1 To 4 Step 1
    RngAddress(i - 1) = RngAddress(i)
Next i
ReDim Preserve RngAddress(1 To 4)
z_RangeAddressAsArray = RngAddress
```

End Function

```
Private Function z_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, Optional Sh
As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Output datatype change from Variant

Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer

'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
Sheets(Sh).Activate

'find column name
On Error GoTo NameExpectedNotExist:
Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole).Select
On Error GoTo 0
'find column index
CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
```

```
'Output
z_GetColumnIndex = ColIndex
Exit Function
```

OptionalArgument:  
Resume Next

```
NameExpectedNotExistent:
    ColIndex = 0
    z_GetColumnIndex = ColIndex
```

```
End Function
```

## **File SmC\_PMEC\_VBA\_OneLinePerResourceReport\_v84**

```
Sub m_start_Mains()
    Dim wbdate As String
    Dim OnlyPIs As Integer

    OnlyPIs = 0 '0=All;1=OnlyPIs
    wbdate = "2012_10_16"

    '*****

    'Call m_Main_DownloadReport_2010Export(OnlyPIs)

    'Call m_Main_GenerateReport

    'Call m_Main_RedFlags(wbdate)

    'Call m_Main_PivotFlat_PiAndTK_CustomizedDataSet_FunctionalReporting(wbdate)
    'Call m_Main_PivotFlat_PiAndTK_CustomizedForPortfolioReporting(wbdate)
    'Call m_Main_PivotFlat_PiAndTK_CustomizedDataSet_OneLinePerId(wbdate)
    'Call m_Main_PivotFlat_Rs_CustomizedDataSet_OneLinePerId(wbdate)

    'Call m_CheckAttributeEntries_PILevel_MappingAndColoring(wbdate)
    'Call m_CheckAttributeEntries_TKLevel_MappingAndColoring(wbdate)
    'runs through if no colored cell exists otherwise a msgbox pops up
    'Call m_ShowTheErrors("Pi")
    'Call m_ShowTheErrors("Tk")

    Call m_Main_ApplyFilteringRules_RemoveFutureCosts_ForPortfolioReporting(wbdate)
    Call m_Main_CurrentDatasetAndBaseline(wbdate)

    Call m_Main_GenerateMilestoneDataSet(wbdate)

    Call m_Main_AllIds_FindRemovedOnes(wbdate)

    Call m_Main_CheckBusinessRules(wbdate)

    Call m_Main_QualCheckGenerateInput(wbdate)
```

Call m\_Main\_GenerateTheRecipientAttachement(wbdate)

Call m\_Main\_GenerateRecipientEmailList(wbdate)

'Call m\_Main\_WorkbooksForTeamspace(wbdate)

End Sub

\*\*\*\*\*

'Replaced by the Word VBA

\*\*\*\*\*

"Manual tasks/settings before starting the macro.

' 'Goto:(\*Preparations before running the macro

"Tools>References

' 'Visual Basic For Applications

' 'Microsoft Excel 12.0 Object Library

' 'OLE Automation

' 'Microsoft Office 12.0 Object Library

' 'Microsoft Forms 2.0 Object Library

' 'Microsoft Visual Basic for Applications Extensibility 5.3c:/Program Files (x86)/Common  
Files/Microsoft Shared/VBA/VBA6/VBE6EXT.OLB

' 'Microsoft IMAPI2 Base Functionality-c:/Windows/system32/imapi2.dll

' 'Microsoft Internet Controls-c:/Windows/SysWOW64/ieframe.dll

,

"(\* Constant initialisation

' Public Const SW\_RESTORE = 9

' Public Const SW\_SHOW = 5

' Public Const SW\_MINIMIZE = 6

")\* Constant initialisation

,

'Private Sub m\_Start\_Main\_DownloadReport\_HTML\_Export()

' 'Activity exported with 1997/2000 Export via HTML

' 'PIs exported with 2007/2010 Export via HTML

' Dim OnlyPIs As Integer

' Dim myPageURL As String

' Dim answer As String

' \*\*\*\*\*

' answer = InputBox("stg or pro", , "pro", 13500, 12500)

' If answer = "stg" Then

' myPageURL = "smartchoice.stg.intra"

' Elseif answer = "pro" Then

' myPageURL = "smartchoice.pro.intra"

' Else

' Stop

' End If

' answer = InputBox("ActivitiesAndPIs = Yes; OnlyPIs = No", , "Yes", 13500, 12500)

' If answer = "Yes" Then

' OnlyPIs = 0 '1=Only PIs, 0=All

' Elseif answer = "No" Then

' OnlyPIs = 1 '1=Only PIs, 0=All

' Else

```

'    Stop
' End If
' Application.Wait (Now + TimeValue("0:00:02"))
' Call m_Main_DownloadReport_HTML_Export(OnlyPIs, myPageURL)
'End Sub
'

'Private Sub m_Start_Main_DownloadPlannedSales_HTML_Export()
' 'Activity exported with 1997/2000 Export via HTML
' 'PIs exported with 2007/2010 Export via HTML
' Dim OnlyPIs As Integer
' Dim myPageURL As String
' Dim answer As String
' *****
' answer = InputBox("stg or pro", , "pro", 13500, 12500)
' If answer = "stg" Then
'     myPageURL = "smartchoice.stg.intra"
' ElseIf answer = "pro" Then
'     myPageURL = "smartchoice.pro.intra"
' Else
'     Stop
' End If
'
' OnlyPIs = 0 '1=Only PIs, 0=All
'
' Application.Wait (Now + TimeValue("0:00:02"))
' Call m_Main_DownloadPlannedSales_HTML_Export(OnlyPIs, myPageURL)
'End Sub
'

'Private Sub m_Main_DownloadReport_HTML_Export(OnlyPIs As Integer, myPageURL As String)
' Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
'     Array("Main_DownloadReport", "Begin", "task: ", CStr(Now()), ""))
'(*Variable declaration
' (*Preparations before running the macro
' Dim OnlyPIs As Integer
' Dim Sh_log As String
' Dim delta1 As Integer
' Dim delta2 As Integer
' Dim mywb As Workbook
' Dim WbPath As String
' Dim wbdate As String
' Dim WbName As String
' (*Create a new Excel Workbook
' (*prepare the SmC window in an IE object
' Dim myIE As SHDocVw.InternetExplorer
' Dim myPageTitle As String
' Dim myPageURL As String
' Dim strWindowTitle As String 'used several times for different names
' Dim My_IE_hWnd As Long
' (*Open the SmC Project-Module
' Dim Target_X0 As Long
' Dim Target_Y0 As Long
' Dim BySyngentaPortfolio As Boolean
' (*ResetButton

```

```

' Dim ResetButtonYes As Integer
' (*Resource Report
' Dim VP_DropdownSecondPos As Long
' Dim VP_Start As Integer
' Dim TotalVPs As Integer
' Dim FailedDownloads_iter As Integer
' Dim VP_iter As Integer
' (**Open the OneLinePerResource Report
' Dim ResourceStyle As Long
' Dim ResourceStyleDelta As Long
' (**Workaround for the selection problem: clicks on the scroll bar
' Dim ScrollClicks_iter As Integer
' Dim ScrollClicks_iter_Max As Integer
' (*Excel Download
' Dim sec As Integer
' Dim DurationSinceDownloadClick As Integer
' Dim secMax_PI As Integer
' Dim My_IE_XL_hWnd As Long
' Dim My_IE_XL_Child_hWnd As Long
' Dim MyChildName As String
' Dim WindowName As String
' Dim WorkbooksEntry1 As String
' Dim WorkbooksEntry2 As String
' (*PiReport
' Dim VP_iter_Pi As Integer
' Dim VP_Start_Pi As Integer
' Dim TotalVPs_Pi As Integer
' Dim PiStyle As Long
' Dim PiStyleDelta As Long
' Dim secMax_Activity As Integer
")*Variable declaration
'

"(*Preparations before running the macro
' 'Manual tasks/settings before starting the macro
' '1. With IE 7 make sure you have only one tab open (the one with SmC)!!!
' '2. Module>Projects Style:Main R&D PI Export_3
' '3. Module>Projects>Open Style: R&D Reporting Master Data Set_5
' '4. Module>Projects>Open Scheduling>Hours&expenditures>One line per resource
' '5. choose only Pi download yes=1 (PiReport) or no=0 (ResourceReport)
' 'default value is 0 (ResourceReport),
' 'if set to 1, then choose the SmC virtual portfolio just before the first PI virtual portfolio
' 'if you set BySyngentaPorfolio=True then you have 15 seconds to set the right VP
' 'OnlyPIs = 0 'default=0,
' '6. choose Smc user rights with and without read only
' 'delta1 = 0 'read only: 0, all rights: 17
' 'delta2 = 0 'read only: 0, all rights: 56
' '7. Calibrate the mouse click on the Open-Module button
' 'Target_X0 = 21 '(calibrate here)
' 'Target_Y0 = 137 '(calibrate here)
' '8. set the path and the name of the workbook that is created (or opened/activated if it already
exists)
' WbPath = "C:\Users\t740698\Desktop\"
' wbdate = z_wbdate(, Now())

```

```

' WbName = "CONFIDENTIAL_SmC_Download" & "_" & wdate & "_V1-0" & ".xlsb"
' 'wbname = "SmC_Download.xlsb"
' '9. Click on the BySyngentaPortfolio-Tab
' 'default value is true (Click)
' 'if set to false, then make sure that the tab in SmC is set to BySyngentaPortfolio
' BySyngentaPortfolio = True 'NoClick=False,Click=True
' '10.Set the Reset-Button for the Resource report
' 'default value is 1
' 'only used in combination with OnlyPis=0 (ResourceReport), no reset with OnlyPis=1 (PiReport)
' ResetButtonYes = 1 'Yes=1, No=0 (does not influence PiReport)
' '11.Resource Report: Number of virtual portfolios
' TotalVPs = 64 '64 if there are 65 (because it starts with 0)
' '12.Resource Style
' 'default=0 (manually set under point 3, no clicking)
' ResourceStyle = 0
' ResourceStyleDelta = 300 'determine the right value
' '13.Max waiting time for the download IE Window
' 'default=35 seconds
' secMax_Activity = 60 '38
' secMax_PI = 220
' '14.After a new SmC release check all positions of buttons that are clicked by the program
' 'find: (new position and mouse click)
' '15.Check from time to time whether the waiting times before new clicks are enough
' '16.Pi Style
' 'default=0 (manually set under point 2, no clicking)
' PiStyle = 0
' PiStyleDelta = 300 'determine the right value
' '17.PiReport: Number of virtual portfolios
' TotalVPs_Pi = 10 '10 if there are 11 Pi virtual portfolios(because it starts with 0)
' '18.IE Zoom must be set to 100% (IE cell in the lower left corner)
' '19.Selection problem workaround
' 'number of clicks on the scroll bar to have all PIs selected
' ScrollClicks_Iter_Max_Big = 9
' ScrollClicks_Iter_Max_Medium = 7
' ScrollClicks_Iter_Max_Small = 5
' ScrollClicks_Iter_Max_VerySmall = 3
' '20.Check that in the one line per resource report all Attributes are visible
' '21.Set the resource report
' 'none=0, OneLinePerResource=1, Gant=2
' 'default=2 since OneLinePerResource sets back the borderline between table and graph
' ResourceReportFlag = 2
' '22.Make sure Outlook Meeting request reminder does not pop up
'
'*)*Preparations before running the macro
'
'(* Create a new Excel Workbook
' 'show the desktop (minimize all windows)
' Call z_ShowDesktop
' Application.Wait (Now + TimeValue("0:00:01"))
' 'add, open or activate an Excel workbook (and session)
' Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, mywb)
' Application.Wait (Now + TimeValue("0:00:03"))
' 'move and resize excel session

```

```

' Call z_ExcelSessionWindowNormal(mywb)
' Call z_ExcelSessionWindowMoveAndResize(mywb, "400", "0", "700", "340")
' Application.Wait (Now + TimeValue("0:00:03"))
' 'minimize all Excel workbooks within the Excel session except mywb
' Call z_ExcelWorkbookWindowMinimizeAll(mywb)
' Call z_ExcelWorkbookWindowNormal(mywb)
' 'do some renaming, deleting and saving
' 'open or activate mywb
' Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, mywb)
' 'bring the Excel session into xlnormal
' Call z_ExcelSessionWindowNormal(mywb)
' Call z_ExcelWorkbookWindowMaximized(mywb)
' 'do the next steps only if they are not already done
' On Error Resume Next
' 'rename logfile
' mywb.Sheets("Sheet1").Select
' Sh_log = "Logfile"
' mywb.Sheets("Sheet1").name = Sh_log
' 'delete empty sheets
' Call z_DeleteWbSheet(mywb, "Sheet2")
' Call z_DeleteWbSheet(mywb, "Sheet3")
' On Error GoTo 0
' 'save workbook
' mywb.Save
'")* Create a new Excel Workbook
'
'(" *prepare the SmC window in an IE object
' 'With IE 7 make sure you have only one tab open (the one with SmC)!!!
' 'or find a solution to toggle between the tabs (send key Ctrl+Tab) until you found the SmC
' 'IE object instantiation
' 'myPageURL = "smartchoice.pro.intra" 'myPageURL = "smartchoice.stg.intra"
' 'Only if not already Open IE: Open IE, Resize, Reposition, Start SmC, Wait 30secs
' myPageTitle = "SmartChoice"
' Set myIE = IE_Preparation(myPageTitle, myPageURL)
' Application.Wait (Now + TimeValue("0:00:01"))
' 'get the IE handle
' My_IE_hWnd = myIE.hwnd
' 'show or restore IE depending on its current state (iconic = minimized)
' If IsIconic(My_IE_hWnd) Then
'   Call ShowWindow(My_IE_hWnd, SW_RESTORE)
' Else
'   Call ShowWindow(My_IE_hWnd, SW_SHOW)
' End If
' 'bring SmC IE to the foreground
' SetForegroundWindow My_IE_hWnd
'")*prepare the SmC window in an IE object
'
' If OnlyPIs = 0 Then
'   Call z_ExcelSessionWindowMinimized(mywb)
'   Call z_ExcelSessionWindowNormal(mywb)
'   notused = InputBox("set the export to 1997-2000", , "no entry", 13500, 12500)
'   If notused <> "no entry" Then
'     Stop

```

```

'    Application.Wait (Now + TimeValue("0:00:02"))
'    End If
'    Application.Wait (Now + TimeValue("0:00:02"))
'    Call SetForegroundWindow(My_IE_hWnd)
'    End If
'
'(* Open the SmC Project-Module
'    Application.Wait (Now + TimeValue("0:00:02"))
'    'Open-Module-Button(new position and mouse click)
'    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
'    Application.Wait (Now + TimeValue("0:00:01"))
'    'Project-Module(new position and mouse click)
'    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
'    Application.Wait (Now + TimeValue("0:00:15"))
'    'BySyngentaPortfolio-Tab(new position and mouse click)
'    If BySyngentaPortfolio Then
'        'new position and mouse click
'        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 144, 123)
'        Application.Wait (Now + TimeValue("0:00:15"))
'    Else
'        Application.Wait (Now + TimeValue("0:00:01"))
'    End If
'*) Open the SmC Project-Module
'
'(*'ResetButton
'    If OnlyPIs = 0 Then
'        If ResetButtonYes Then
'            'VirtualPortfolioAdmin-Button(new position and mouse click)
'            Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 238 + delta1, 63)
'            Application.Wait (Now + TimeValue("0:00:01"))
'            'Reset-Button(new position and mouse click)
'            Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 238 + delta1, 148)
'            Application.Wait (Now + TimeValue("0:00:15"))
'        Else
'            Application.Wait (Now + TimeValue("0:00:01"))
'        End If
'    Else
'        'Select the last activity virtual portfolio, so that the programm starts with
'        'selecting the next virtual portfolio which is the first Pi virtual portfolio
'    End If
'*)'ResetButton
'
'    If OnlyPIs = 0 Then
'        Call z_ExcelSessionWindowMinimized(mywb)
'        Call z_ExcelSessionWindowNormal(mywb)
'        notused = InputBox("set the style to: return to default style on PI report (faster selection)", , "no
entry", 13500, 12500)
'        If notused <> "no entry" Then
'            Stop
'            Application.Wait (Now + TimeValue("0:00:02"))
'        End If
'        Application.Wait (Now + TimeValue("0:00:02"))
'        Call SetForegroundWindow(My_IE_hWnd)

```



```

' End If
'
'(*Resource Report
' Application.Wait (Now + TimeValue("0:00:02"))
' If OnlyPIs = 0 Then
'     FailedDownloads_iter = 1
'     VP_Start = 0 '0
'     'loop over the resource VPs
'     For VP_iter = VP_Start To TotalVPs
'
'         '(**Bring IE to the foreground
'         'show or restore IE depending on its current state
'         If IsIconic(My_IE_hWnd) Then
'             Call ShowWindow(My_IE_hWnd, SW_RESTORE)
'         Else
'             Call ShowWindow(My_IE_hWnd, SW_SHOW)
'         End If
'         'bring SmC IE to the foreground
'         Call SetForegroundWindow(My_IE_hWnd)
'         'To be sure SmC is in an idle state before going back to the portfolio module
'         Application.Wait (Now + TimeValue("0:00:03"))
'     '**Bring IE to the foreground
'
'     '(**close the activities or not and bring SmC IE back to the modul "project"
'     If VP_iter <> 0 Then
'         'Open-Module-Button(new position and mouse click)
'         Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
'         Application.Wait (Now + TimeValue("0:00:03"))
'         'Project-Module(new position and mouse click)
'         Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
'         Application.Wait (Now + TimeValue("0:00:15"))
'     End If
'     '**close the activities or not and bring SmC IE back to the modul "project"
'
'     '(**Choose a new Virtual Portfolio
'     'VirtualPortfolioChoice-DropDown(new position and mouse click)
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 533 + delta1, 63)
'     Application.Wait (Now + TimeValue("0:00:05"))
'     'VirtualPortfolioChoice(new position and mouse click)
'     VP_DropdownSecondPos = 97 'Delta Y0 to click on the second VP
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 338 + delta1,
VP_DropdownSecondPos)
'     Application.Wait (Now + TimeValue("0:00:15"))
'     '**Choose a new Virtual Portfolio
'
'     '(**Workaround for the selection problem: clicks on the scroll bar
'     '0 means VP 0+1
'     '1 means VP 1+1
'     '2 means VP 2+1
'     'Nof PIs from 140-200 as well as the last one: VP_iter = 58 (move it to the right place if it
becomes the second before last)
'     If VP_iter = 19 - 1 Or _
'         VP_iter = 34 - 1 Or _

```

```

'      VP_iter = 36 - 1 Or _
'      VP_iter = 39 - 1 Or _
'      VP_iter = 54 - 1 Or _
'      VP_iter = 63 - 1 Then
'      'SelectAll-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Big
'          'Scroll-Bar(new position and mouse click)
'          '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'          Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'          Application.Wait (Now + TimeValue("0:00:02"))
'          'SelectAll-Button(new position and mouse click)
'          Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'          Application.Wait (Now + TimeValue("0:00:01"))
'      Next
'      'Nof PIs from 100-139
'      Elseif VP_iter = 4 - 1 Or _
'          VP_iter = 18 - 1 Or _
'          VP_iter = 20 - 1 Or _
'          VP_iter = 21 - 1 Or _
'          VP_iter = 37 - 1 Or _
'          VP_iter = 38 - 1 Or _
'          VP_iter = 40 - 1 Or _
'          VP_iter = 42 - 1 Or _
'          VP_iter = 43 - 1 Or _
'          VP_iter = 44 - 1 Or _
'          VP_iter = 45 - 1 Or _
'          VP_iter = 47 - 1 Or _
'          VP_iter = 49 - 1 Or _
'          VP_iter = 53 - 1 Or _
'          VP_iter = 56 - 1 Or _
'          VP_iter = 58 - 1 Or _
'          VP_iter = 61 - 1 Or _
'          VP_iter = 62 - 1 Then
'          'SelectAll-Button(new position and mouse click)
'          Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'          Application.Wait (Now + TimeValue("0:00:01"))
'          For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Medium
'              'Scroll-Bar(new position and mouse click)
'              '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'              Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'              Application.Wait (Now + TimeValue("0:00:02"))
'              'SelectAll-Button(new position and mouse click)
'              Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'              Application.Wait (Now + TimeValue("0:00:01"))
'          Next
'          'Nof PIs from 60-99
'          Elseif VP_iter = 2 - 1 Or _
'              VP_iter = 3 - 1 Or _
'              VP_iter = 6 - 1 Or _

```

```

'      VP_iter = 22 - 1 Or _
'      VP_iter = 23 - 1 Or _
'      VP_iter = 29 - 1 Or _
'      VP_iter = 32 - 1 Or _
'      VP_iter = 33 - 1 Or _
'      VP_iter = 41 - 1 Or _
'      VP_iter = 46 - 1 Or _
'      VP_iter = 48 - 1 Or _
'      VP_iter = 50 - 1 Or _
'      VP_iter = 51 - 1 Or _
'      VP_iter = 55 - 1 Or _
'      VP_iter = 59 - 1 Or _
'      VP_iter = 60 - 1 Then
'      'SelectAll-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Small
'          'Scroll-Bar(new position and mouse click)
'          '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'          Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'          Application.Wait (Now + TimeValue("0:00:02"))
'          'SelectAll-Button(new position and mouse click)
'          Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'          Application.Wait (Now + TimeValue("0:00:01"))
'      Next
'      'Nof PIs from 1-59
'      Else
'          For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_VerySmall
'              'Scroll-Bar(new position and mouse click)
'              '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'              Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'              Application.Wait (Now + TimeValue("0:00:02"))
'              'SelectAll-Button(new position and mouse click)
'              Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'              Application.Wait (Now + TimeValue("0:00:01"))
'          Next
'      End If
'      '**Workaround for the selection problem: clicks on the scroll bar
'
'      '**Open the OneLinePerResource or Gant Report
'      'SelectAll-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'      Application.Wait (Now + TimeValue("0:00:05"))
'      'Open-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 120 + delta2, 64)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      'OpenSelectedProjects-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 131 + delta2, 91)
'      Application.Wait (Now + TimeValue("0:00:15"))
'      'Choose the one line per resource report or the gant report in the first loop
'      If VP_iter = 0 Then

```

```

'
' If ResourceReportFlag = 1 Then
'     'Scheduling-Button(new position and mouse click)
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 29, 30)
'     Application.Wait (Now + TimeValue("0:00:03"))
'     'Hours&Expenditures-DropDownEntry(new position no click!)
'     Call z_SetMousePos(Target_X0, Target_Y0, 29, 163)
'     Application.Wait (Now + TimeValue("0:00:03"))
'     'OneLinePerResource-DropDownEntry(new position and mouse click)
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 239, 163)
'     Application.Wait (Now + TimeValue("0:00:10"))
' Elseif ResourceReportFlag = 2 Then
'     'Scheduling-Button(new position and mouse click)
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 29, 30)
'     Application.Wait (Now + TimeValue("0:00:03"))
'     'Gant-DropDownEntry(new position and mouse click)
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 29, 63)
'     Application.Wait (Now + TimeValue("0:00:10"))
' End If
' End If
' 'select the "R&D Reporting Master Data Set" style
' If ResourceStyle Then
'     'Style-Button(new position and mouse click)
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 676, 135)
'     Application.Wait (Now + TimeValue("0:00:03"))
'     'Style-Choice(new position and mouse click)
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 580, ResourceStyleDelta)
'     Application.Wait (Now + TimeValue("0:00:15"))
' End If
' '**Open the OneLinePerResource or Gant Report
'
' If VP_iter = 0 Then
'     Call z_ExcelSessionWindowMinimized(mywb)
'     Call z_ExcelSessionWindowNormal(mywb)
'     notused = InputBox("set the style to: R&D Reporting Master DataSet_6 for the Activity
download", , "no entry", 13500, 12500)
'     If notused <> "no entry" Then
'         Stop
'         Application.Wait (Now + TimeValue("0:00:02"))
'     End If
'     Application.Wait (Now + TimeValue("0:00:02"))
'     Call SetForegroundWindow(My_IE_hWnd)
' End If
'
' '**Excel Download
'     '**Excel-Download-Button(new position and mouse click)
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1025, 0)
'     Application.Wait (Now + TimeValue("0:00:02"))
'     '**Minimize SmC IE
'     Call ShowWindow(My_IE_hWnd, SW_MINIMIZE)
'     DoEvents
'     Application.Wait (Now + TimeValue("0:00:01"))
'     '**make excel active
'     Call z_ExcelSessionWindowNormal(mywb)

```

```

'      Call z_ExcelWorkbookWindowNormal(mywb)
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:01"))
'      ***Determine when the Download IE appears
'      'if the time secMax is not exceeded, the download should work out
'      'if the time secMax is exceeded, the download is about to fail, wait an try to close the IE
window
'      'that appears after secMax. After some time this window gets the name HTTP 500 ...
'      strWindowTitle = "Windows Internet Explorer provided by Syngenta"
'      sec = z_IE_WaitUntilNewWindowExists_HTML_Export(strWindowTitle, secMax_Activity)
'      DoEvents
'      ***If IE download fails (HTTP 500 ...), wait and close IE
'      'to be sure SmC is in an idle state before going back to the portfolio module
'      If sec > secMax_Activity Or sec = secMax_Activity Then
'          'wait until SmC is in an idle state
'          Application.Wait (Now + TimeValue("0:02:00")) 'maybe enhance to 10 minutes
'          'try to close the IE window
'          strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided
by Syngenta"
'          Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
'          End If
'      ***minimize IE that prepares the download and wait until the IE download is exported into
Excel
'      strWindowTitle = "Windows Internet Explorer provided by Syngenta"
'      My_IE_XL_hWnd = z_FindWindowHandle(strWindowTitle)
'      If My_IE_XL_hWnd <> 0 Then
'          'minimize IE
'          Call ShowWindow(My_IE_XL_hWnd, SW_MINIMIZE)
'          'give some waiting time until the IE download is exported into Excel
'          DoEvents
'          Application.Wait (Now + TimeValue("0:00:10"))
'          End If
'      ***write out the window name
'      WindowName = Workbooks(Workbooks.count).name
'      WorkbooksEntry1 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 5)
'      WorkbooksEntry2 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 6)
'      mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 5) = VP_iter + 1
'      mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 6) = sec
'      mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 7) = WindowName
'      mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 8) = WorkbooksEntry1
'      mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 9) = WorkbooksEntry2
'      ***Move the newly created download Wb into the SmC_Download file
'      Call z_MoveWbSheetsIntoAnotherWb_V2(mywb, "SmC_A_VP_", VP_iter + 1)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      ***minimize all Excel workbooks within the Excel session
'      Call z_ExcelWorkbookWindowMinimizeAll(mywb)
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:01"))
'      ***If IE fails, or IE failed to close before, so try to close the window now
'      strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet
Explorer provided by Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
'      Application.Wait (Now + TimeValue("0:00:01"))

```

```

'      strWindowTitle = "Verify - Windows Internet Explorer provided by Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided by
Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
'      Application.Wait (Now + TimeValue("0:00:01"))
'  '**Excel Download
'  Next VP_iter
'  'save workbook
'  mywb.Save
'  DoEvents
'  Application.Wait (Now + TimeValue("0:00:10"))
' End If
')*Resource Report
'
'
'(*PiReport
'  '(*close the activities or not and bring SmC IE back to the modul "project"
'  If OnlyPIs = 0 Then
'    'Open-Module-Button(new position and mouse click)
'    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
'    Application.Wait (Now + TimeValue("0:00:03"))
'    'Project-Module(new position and mouse click)
'    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
'    Application.Wait (Now + TimeValue("0:00:14"))
'  End If
'  ')*close the activities or not and bring SmC IE back to the modul "project"
'
'  If OnlyPIs = 0 Or OnlyPIs = 1 Then
'    Call z_ExcelSessionWindowMinimized(mywb)
'    Call z_ExcelSessionWindowNormal(mywb)
'    notused = InputBox("set the export to 2007-2010 for the PI download", , "no entry", 13500,
12500)
'    If notused <> "no entry" Then
'      Stop
'      Application.Wait (Now + TimeValue("0:00:02"))
'    End If
'    notused = InputBox("set the style to Main R&D PI Export_5 for the PI download", , "no entry",
13500, 12500)
'    If notused <> "no entry" Then
'      Stop
'      Application.Wait (Now + TimeValue("0:00:02"))
'    End If
'    notused = InputBox("set the VP before the first PI VP", , "no entry", 13500, 12500)
'    If notused <> "no entry" Then
'      Stop
'      Application.Wait (Now + TimeValue("0:00:02"))
'    End If
'    Application.Wait (Now + TimeValue("0:00:02"))
'    Call SetForegroundWindow(My_IE_hWnd)
'  End If
'

```

```

' If OnlyPIs = 0 Or OnlyPIs = 1 Then
'   'loop over all Pi virtual portfolios
'   VP_Start_Pi = 0
'   For VP_iter_Pi = VP_Start_Pi To TotalVPs_Pi
'   '
'   '(**Bring IE to the foreground
'   '   'show or restore IE depending on its current state
'   '   If IsIconic(My_IE_hWnd) Then
'   '       Call ShowWindow(My_IE_hWnd, SW_RESTORE)
'   '   Else
'   '       Call ShowWindow(My_IE_hWnd, SW_SHOW)
'   '   End If
'   '   'bring SmC IE to the foreground
'   '   Call SetForegroundWindow(My_IE_hWnd)
'   '   'To be sure SmC is in an idle state before going back to the portfolio module
'   '   Application.Wait (Now + TimeValue("0:00:05"))
'   '(**Bring IE to the foreground
'   '
'
"   '(**close the activities or not and bring SmC IE back to the modul "project"
"   '   If VP_iter_Pi = VP_Start_Pi Then
"   '       'Open-Module-Button(new position and mouse click)
"   '       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
"   '       Application.Wait (Now + TimeValue("0:00:01"))
"   '       'Project-Module(new position and mouse click)
"   '       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
"   '       Application.Wait (Now + TimeValue("0:00:14"))
"   '   End If
"   '(**close the activities or not and bring SmC IE back to the modul "project"
'   '
'
'   '(**select the "Main R&D PI Report" style
'   '   If PiStyle Then
'   '       'Style-Button(new position and mouse click)
'   '       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1157, 141)
'   '       Application.Wait (Now + TimeValue("0:00:03"))
'   '       'Style-Choice(new position and mouse click)
'   '       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1096, PiStyleDelta)
'   '       Application.Wait (Now + TimeValue("0:00:10"))
'   '   End If
'   '(**select the "Main R&D PI Report" style
'   '
'
'   '(**Choose a new Virtual Portfolio
'   '   'VirtualPortfolioChoice-DropDown(new position and mouse click)
'   '   Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 533 + delta1, 63)
'   '   Application.Wait (Now + TimeValue("0:00:05"))
'   '   'VirtualPortfolioChoice(new position and mouse click)
'   '   VP_DropdownSecondPos = 97 'Delta Y0 to click on the second VP
'   '   Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 338 + delta1,
VP_DropdownSecondPos)
'   '   Application.Wait (Now + TimeValue("0:00:15"))
'   '(**Choose a new Virtual Portfolio
'   '
'
'   '(**Excel Download
'   '   '***Excel-Download-Button(new position and mouse click)

```

```

'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1025, 0)
'      Application.Wait (Now + TimeValue("0:00:02"))
'      ***Minimize SmC IE
'      Call ShowWindow(My_IE_hWnd, SW_MINIMIZE)
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:01"))
'      ***make excel active
'      Call z_ExcelSessionWindowNormal(mywb)
'      Call z_ExcelWorkbookWindowNormal(mywb)
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:01"))
'      ***Determine when the Download IE appears
'      'if the time secMax is not exceeded, the download should work out
'      'if the time secMax is exceeded, the download is about to fail, wait an try to close the IE
window
'      'that appears after secMax. After some time this window gets the name HTTP 500 ...
'      DurationSinceDownloadClick = 4
'      strWindowTitle = "Windows Internet Explorer provided by Syngenta"
'      sec = z_IE_WaitUntilNewWindowExists_HTML_Export(strWindowTitle, secMax_PI -
DurationSinceDownloadClick)
'      sec = sec + DurationSinceDownloadClick
'      DoEvents
'
'      ***If IE download fails (HTTP 500 ...), wait and close IE
'      'to be sure SmC is in an idle state before going back to the portfolio module
'      If sec > secMax_PI Or sec = secMax_PI Then
'      'wait until SmC is in an idle state
'      Application.Wait (Now + TimeValue("0:02:00")) 'maybe enhance to 10 minutes
'      'try to close the IE window
'      strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided
by Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter_Pi, FailedDownloads_iter)
'      End If
'      ***minimize IE that prepares the download and wait until the IE download is exported into
Excel
'      strWindowTitle = "Windows Internet Explorer provided by Syngenta"
'      My_IE_XL_hWnd = z_FindWindowHandle(strWindowTitle)
'      If My_IE_XL_hWnd <> 0 Then
'      'minimize IE
'      Call ShowWindow(My_IE_XL_hWnd, SW_MINIMIZE)
'      'give some waiting time until the IE download is exported into Excel
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:10"))
'      End If
'      ***write out the window name
'      WindowName = Workbooks(Workbooks.count).name
'      WorkbooksEntry1 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 5)
'      WorkbooksEntry2 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 6)
'      mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 5) = VP_iter_Pi + 1
'      mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 6) = sec
'      mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 7) = WindowName
'      mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 8) = WorkbooksEntry1
'      mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 9) = WorkbooksEntry2

```



```

'      ***Save the export workbook
'      Application.DisplayAlerts = False
'      Call z_WorkbookSave(ActiveWorkbook)
'      Application.DisplayAlerts = True
'      ***Move newly created download Wb into the SmC_Download file
'      Call z_MoveWbSheetsIntoAnotherWb_V2(mywb, "SmC_P_VP_", VP_iter_Pi + 1)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      ***Save the download workbook
'      Call z_WorkbookSave(mywb)
'      ***minimize all Excel workbooks within the Excel session
'      Call z_ExcelWorkbookWindowMinimizeAll(mywb)
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:01"))
'      ***If IE fails, or IE failed to close before, so try to close the window now
'      strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet
Explorer provided by Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter_Pi, FailedDownloads_iter)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided by
Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter_Pi, FailedDownloads_iter)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      '(*Excel Download
'      Next VP_iter_Pi
'      'save workbook
'      mywb.Save
'      End If
'(*PiReport
'      notused = InputBox("Download has been finished", , "no entry", 13500, 12500)
'      Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
'      Array("Main_DownloadReport", "End", "task: ", CStr(Now()), ""))
'End Sub
'
'
'
'Private Sub m_Main_DownloadPlannedSales_HTML_Export(OnlyPIs As Integer, myPageURL As
String)
'      Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
'      Array("Main_DownloadReport", "Begin", "task: ", CStr(Now()), ""))
'(*Variable declaration
'      '(*Preparations before running the macro
'      'Dim OnlyPIs As Integer
'      Dim Sh_log As String
'      Dim delta1 As Integer
'      Dim delta2 As Integer
'      Dim mywb As Workbook
'      Dim WbPath As String
'      Dim wbdate As String
'      Dim WbName As String
'      '(*Create a new Excel Workbook
'      '(*prepare the SmC window in an IE object
'      Dim myIE As SHDocVw.InternetExplorer
'      Dim myPageTitle As String

```

```

' Dim myPageURL As String
' Dim strWindowTitle As String 'used several times for different names
' Dim My_IE_hWnd As Long
' (*Open the SmC Project-Module
' Dim Target_X0 As Long
' Dim Target_Y0 As Long
' Dim BySyngentaPortfolio As Boolean
' (*ResetButton
' Dim ResetButtonYes As Integer
' (*Resource Report
' Dim VP_DropdownSecondPos As Long
' Dim VP_Start As Integer
' Dim TotalVPs As Integer
' Dim FailedDownloads_iter As Integer
' Dim VP_iter As Integer
' (**Open the OneLinePerResource Report
' Dim ResourceStyle As Long
' Dim ResourceStyleDelta As Long
' (**Workaround for the selection problem: clicks on the scroll bar
' Dim ScrollClicks_iter As Integer
' Dim ScrollClicks_iter_Max As Integer
' (**Excel Download
' Dim sec As Integer
' Dim DurationSinceDownloadClick As Integer
' Dim secMax_Pi As Integer
' Dim My_IE_XL_hWnd As Long
' Dim My_IE_XL_Child_hWnd As Long
' Dim MyChildName As String
' Dim WindowName As String
' Dim WorkbooksEntry1 As String
' Dim WorkbooksEntry2 As String
' (*PiReport
' Dim VP_iter_Pi As Integer
' Dim VP_Start_Pi As Integer
' Dim TotalVPs_Pi As Integer
' Dim PiStyle As Long
' Dim PiStyleDelta As Long
' Dim secMax_Activity As Integer
")*Variable declaration
'

"(*Preparations before running the macro
' 'Manual tasks/settings before starting the macro
' '1. With IE 7 make sure you have only one tab open (the one with SmC)!!!
' '2. Module>Projects Style:Main R&D PI Export_3
' '3. Module>Projects>Open Style: R&D Reporting Master Data Set_5
' '4. Module>Projects>Open Scheduling>Hours&expenditures>One line per resource
' '5. choose only Pi download yes=1 (PiReport) or no=0 (ResourceReport)
' 'default value is 0 (ResourceReport),
' 'if set to 1, then choose the SmC virtual portfolio just before the first PI virtual portfolio
' 'if you set BySyngentaPorfolio=True then you have 15 seconds to set the right VP
' 'OnlyPis = 0 'default=0,
' '6. choose Smc user rights with and without read only
' delta1 = 0 'read only: 0, all rights: 17

```

```

'    delta2 = 0 'read only: 0, all rights: 56
' '7. Calibrate the mouse click on the Open-Module button
'    Target_X0 = 21 '(calibrate here)
'    Target_Y0 = 137 '(calibrate here)
' '8. set the path and the name of the workbook that is created (or opened/activated if it already
exists)
'    WbPath = "C:\Users\t740698\Desktop\"
'    wbdate = z_wbdate(, Now())
'    WbName = "CONFIDENTIAL_SmC_DownloadPlannedSales" & "_" & wbdate & "_V1-0" & ".xlsb"
'    'wbname = "SmC_Download.xlsb"
' '9. Click on the BySyngentaPortfolio-Tab
'    'default value is true (Click)
'    'if set to false, then make sure that the tab in SmC is set to BySyngentaPortfolio
'    BySyngentaPortfolio = True 'NoClick=False,Click=True
' '10.Set the Reset-Button for the Resource report
'    'default value is 1
'    'only used in combination with OnlyPis=0 (ResourceReport), no reset with OnlyPis=1 (PiReport)
'    ResetButtonYes = 1 'Yes=1, No=0 (does not influence PiReport)
' '11.Resource Report: Number of virtual portfolios
'    TotalVPs = 57 '57 if there are 58 (because it starts with 0)
' '12.Resource Style
'    'default=0 (manually set under point 3, no clicking)
'    ResourceStyle = 0
'    ResourceStyleDelta = 300 'determine the right value
' '13.Max waiting time for the download IE Window
'    'default=35 seconds
'    secMax_Activity = 60 '38
'    secMax_PI = 220
' '14.After a new SmC release check all positions of buttons that are clicked by the program
'    'find: (new position and mouse click)
' '15.Check from time to time whether the waiting times before new clicks are enough
' '16.Pi Style
'    'default=0 (manually set under point 2, no clicking)
'    PiStyle = 0
'    PiStyleDelta = 300 'determine the right value
' '17.PiReport: Number of virtual portfolios
'    TotalVPs_Pi = 9 '9 if there are 10 Pi virtual portfolios(because it starts with 0)
' '18.IE Zoom must be set to 100% (IE cell in the lower left corner)
' '19.Selection problem workaround
'    'number of clicks on the scroll bar to have all PIs selected
'    ScrollClicks_Iter_Max_Big = 9
'    ScrollClicks_Iter_Max_Medium = 7
'    ScrollClicks_Iter_Max_Small = 5
'    ScrollClicks_Iter_Max_VerySmall = 3
' '20.Check that in the one line per resource report all Attributes are visible
' '21.Set the resource report
'    'none=0, OneLinePerResource=1, Gant=2
'    'default=2 since OneLinePerResource sets back the borderline between table and graph
'    ResourceReportFlag = 3
' '22.Make sure Outlook Meeting request reminder does not pop up
'
'*)*Preparations before running the macro
'

```

```

"(* Create a new Excel Workbook
' show the desktop (minimize all windows)
' Call z_ShowDesktop
' Application.Wait (Now + TimeValue("0:00:01"))
' add, open or activate an Excel workbook (and session)
' Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, mywb)
' Application.Wait (Now + TimeValue("0:00:03"))
' move and resize excel session
' Call z_ExcelSessionWindowNormal(mywb)
' Call z_ExcelSessionWindowMoveAndResize(mywb, "400", "0", "700", "340")
' Application.Wait (Now + TimeValue("0:00:03"))
' minimize all Excel workbooks within the Excel session except mywb
' Call z_ExcelWorkbookWindowMinimizeAll(mywb)
' Call z_ExcelWorkbookWindowNormal(mywb)
' do some renaming, deleting and saving
' open or activate mywb
' Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, mywb)
' bring the Excel session into xlnormal
' Call z_ExcelSessionWindowNormal(mywb)
' Call z_ExcelWorkbookWindowMaximized(mywb)
' do the next steps only if they are not already done
' On Error Resume Next
' rename logfile
' mywb.Sheets("Sheet1").Select
' Sh_log = "Logfile"
' mywb.Sheets("Sheet1").name = Sh_log
' delete empty sheets
' Call z_DeleteWbSheet(mywb, "Sheet2")
' Call z_DeleteWbSheet(mywb, "Sheet3")
' On Error GoTo 0
' save workbook
' mywb.Save
")* Create a new Excel Workbook
,

"(* prepare the SmC window in an IE object
' With IE 7 make sure you have only one tab open (the one with SmC)!!!
' or find a solution to toggle between the tabs (send key Ctrl+Tab) until you found the SmC
' IE object instantiation
' myPageURL = "smartchoice.pro.intra" 'myPageURL = "smartchoice.stg.intra"
' Only if not already Open IE: Open IE, Resize, Reposition, Start SmC, Wait 30secs
' myPageTitle = "SmartChoice"
' Set myIE = IE_Preparation(myPageTitle, myPageURL)
' Application.Wait (Now + TimeValue("0:00:01"))
' get the IE handle
' My_IE_hWnd = myIE.hwnd
' show or restore IE depending on its current state (iconic = minimized)
' If IsIconic(My_IE_hWnd) Then
'     Call ShowWindow(My_IE_hWnd, SW_RESTORE)
' Else
'     Call ShowWindow(My_IE_hWnd, SW_SHOW)
' End If
' bring SmC IE to the foreground
' SetForegroundWindow My_IE_hWnd

```

```

")*prepare the SmC window in an IE object
'
' If OnlyPIs = 0 Then
'   Call z_ExcelSessionWindowMinimized(mywb)
'   Call z_ExcelSessionWindowNormal(mywb)
'   notused = InputBox("set the export to 1997-2000", , "no entry", 13500, 12500)
'   If notused <> "no entry" Then
'       Stop
'       Application.Wait (Now + TimeValue("0:00:02"))
'   End If
'   Application.Wait (Now + TimeValue("0:00:02"))
'   Call SetForegroundWindow(My_IE_hWnd)
' End If
'
"(*Open the SmC Project-Module
' Application.Wait (Now + TimeValue("0:00:02"))
' 'Open-Module-Button(new position and mouse click)
' Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
' Application.Wait (Now + TimeValue("0:00:01"))
' 'Project-Module(new position and mouse click)
' Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
' Application.Wait (Now + TimeValue("0:00:15"))
' 'BySyngentaPortfolio-Tab(new position and mouse click)
' If BySyngentaPortfolio Then
'   'new position and mouse click
'   Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 144, 123)
'   Application.Wait (Now + TimeValue("0:00:15"))
' Else
'   Application.Wait (Now + TimeValue("0:00:01"))
' End If
")*Open the SmC Project-Module
'
"(*'ResetButton
' If OnlyPIs = 0 Then
'   If ResetButtonYes Then
'       'VirtualPortfolioAdmin-Button(new position and mouse click)
'       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 238 + delta1, 63)
'       Application.Wait (Now + TimeValue("0:00:01"))
'       'Reset-Button(new position and mouse click)
'       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 238 + delta1, 148)
'       Application.Wait (Now + TimeValue("0:00:15"))
'   Else
'       Application.Wait (Now + TimeValue("0:00:01"))
'   End If
' Else
'   'Select the last activity virtual portfolio, so that the programm starts with
'   'selecting the next virtual portfolio which is the first Pi virtual portfolio
' End If
")*'ResetButton
'
' If OnlyPIs = 0 Then
'   Call z_ExcelSessionWindowMinimized(mywb)
'   Call z_ExcelSessionWindowNormal(mywb)

```

```

'    notused = InputBox("set the style to: return to default style on PI report (faster selection)", , "no
entry", 13500, 12500)
'    If notused <> "no entry" Then
'        Stop
'        Application.Wait (Now + TimeValue("0:00:02"))
'    End If
'    Application.Wait (Now + TimeValue("0:00:02"))
'    Call SetForegroundWindow(My_IE_hWnd)
' End If
'
'(*Resource Report
' Application.Wait (Now + TimeValue("0:00:02"))
' If OnlyPIs = 0 Then
'     FailedDownloads_iter = 1
'     VP_Start = 0 '0
'     'loop over the resource VPs
'     For VP_iter = VP_Start To TotalVPs
'
'         '(*Bring IE to the foreground
'         'show or restore IE depending on its current state
'         If IsIconic(My_IE_hWnd) Then
'             Call ShowWindow(My_IE_hWnd, SW_RESTORE)
'         Else
'             Call ShowWindow(My_IE_hWnd, SW_SHOW)
'         End If
'         'bring SmC IE to the foreground
'         Call SetForegroundWindow(My_IE_hWnd)
'         'To be sure SmC is in an idle state before going back to the portfolio module
'         Application.Wait (Now + TimeValue("0:00:03"))
'     ')*Bring IE to the foreground
'
'     '(*close the activities or not and bring SmC IE back to the modul "project"
'     If VP_iter <> 0 Then
'         'Open-Module-Button(new position and mouse click)
'         Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
'         Application.Wait (Now + TimeValue("0:00:03"))
'         'Project-Module(new position and mouse click)
'         Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
'         Application.Wait (Now + TimeValue("0:00:15"))
'     End If
'     ')*close the activities or not and bring SmC IE back to the modul "project"
'
'     '(*Choose a new Virtual Portfolio
'     'VirtualPortfolioChoice-DropDown(new position and mouse click)
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 533 + delta1, 63)
'     Application.Wait (Now + TimeValue("0:00:05"))
'     'VirtualPortfolioChoice(new position and mouse click)
'     VP_DropdownSecondPos = 97 'Delta Y0 to click on the second VP
'     Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 338 + delta1,
VP_DropdownSecondPos)
'     Application.Wait (Now + TimeValue("0:00:15"))
'     ')*Choose a new Virtual Portfolio
'

```

```

'  '(**Workaround for the selection problem: clicks on the scroll bar
'    '0 means VP 0+1
'    '1 means VP 1+1
'    '2 means VP 2+1
'    'Nof PIs from 140-200 as well as the last one: VP_iter = 58 (move it to the right place if it
becomes the second before last)
'    If VP_iter = 19 - 1 Or _
'      VP_iter = 34 - 1 Or _
'      VP_iter = 36 - 1 Or _
'      VP_iter = 39 - 1 Or _
'      VP_iter = 54 - 1 Or _
'      VP_iter = 63 - 1 Then
'        'SelectAll-Button(new position and mouse click)
'        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'        Application.Wait (Now + TimeValue("0:00:01"))
'        For ScrollClicks_iter = 1 To ScrollClicks_iter_Max_Big
'          'Scroll-Bar(new position and mouse click)
'          '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'            Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'            Application.Wait (Now + TimeValue("0:00:02"))
'            'SelectAll-Button(new position and mouse click)
'            Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'            Application.Wait (Now + TimeValue("0:00:01"))
'        Next
'    'Nof PIs from 100-139
'    ElseIf VP_iter = 4 - 1 Or _
'      VP_iter = 18 - 1 Or _
'      VP_iter = 20 - 1 Or _
'      VP_iter = 21 - 1 Or _
'      VP_iter = 37 - 1 Or _
'      VP_iter = 38 - 1 Or _
'      VP_iter = 40 - 1 Or _
'      VP_iter = 42 - 1 Or _
'      VP_iter = 43 - 1 Or _
'      VP_iter = 44 - 1 Or _
'      VP_iter = 45 - 1 Or _
'      VP_iter = 47 - 1 Or _
'      VP_iter = 49 - 1 Or _
'      VP_iter = 53 - 1 Or _
'      VP_iter = 56 - 1 Or _
'      VP_iter = 58 - 1 Or _
'      VP_iter = 61 - 1 Or _
'      VP_iter = 62 - 1 Then
'        'SelectAll-Button(new position and mouse click)
'        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'        Application.Wait (Now + TimeValue("0:00:01"))
'        For ScrollClicks_iter = 1 To ScrollClicks_iter_Max_Medium
'          'Scroll-Bar(new position and mouse click)
'          '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'            Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'            Application.Wait (Now + TimeValue("0:00:02"))

```

```

'      'SelectAll-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      Next
'      'Nof Pls from 60-99
'      Elseif VP_iter = 2 - 1 Or _
'      VP_iter = 3 - 1 Or _
'      VP_iter = 6 - 1 Or _
'      VP_iter = 22 - 1 Or _
'      VP_iter = 23 - 1 Or _
'      VP_iter = 29 - 1 Or _
'      VP_iter = 32 - 1 Or _
'      VP_iter = 33 - 1 Or _
'      VP_iter = 41 - 1 Or _
'      VP_iter = 46 - 1 Or _
'      VP_iter = 48 - 1 Or _
'      VP_iter = 50 - 1 Or _
'      VP_iter = 51 - 1 Or _
'      VP_iter = 55 - 1 Or _
'      VP_iter = 59 - 1 Or _
'      VP_iter = 60 - 1 Then
'      'SelectAll-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Small
'      'Scroll-Bar(new position and mouse click)
'      '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'      Application.Wait (Now + TimeValue("0:00:02"))
'      'SelectAll-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      Next
'      'Nof Pls from 1-59
'      Else
'      For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_VerySmall
'      'Scroll-Bar(new position and mouse click)
'      '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'      Application.Wait (Now + TimeValue("0:00:02"))
'      'SelectAll-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      Next
'      End If
'      ')*Workaround for the selection problem: clicks on the scroll bar
'
'      '(**Open the OneLinePerResource or Gant Report
'      'SelectAll-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'      Application.Wait (Now + TimeValue("0:00:05"))

```



```

'
'Open-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 120 + delta2, 64)
Application.Wait (Now + TimeValue("0:00:01"))
'
'OpenSelectedProjects-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 131 + delta2, 91)
Application.Wait (Now + TimeValue("0:00:15"))
'
'Choose the one line per resource report or the gant report in the first loop
If VP_iter = 0 Then
    If ResourceReportFlag = 1 Then
        'Scheduling-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 29, 30)
        Application.Wait (Now + TimeValue("0:00:03"))
        'Hours&Expenditures-DropDownEntry(new position no click!)
        Call z_SetMousePos(Target_X0, Target_Y0, 29, 163)
        Application.Wait (Now + TimeValue("0:00:03"))
        'OneLinePerResource-DropDownEntry(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 239, 163)
        Application.Wait (Now + TimeValue("0:00:10"))
    ElseIf ResourceReportFlag = 2 Then
        'Scheduling-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 29, 30)
        Application.Wait (Now + TimeValue("0:00:03"))
        'Gant-DropDownEntry(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 29, 63)
        Application.Wait (Now + TimeValue("0:00:10"))
    ElseIf ResourceReportFlag = 3 Then
        'Tracking-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 180, 30)
        Application.Wait (Now + TimeValue("0:00:03"))
        'ViewDetailedCosts-DropDownEntry(new position and mouse click)
        Call z_SetMousePos(Target_X0, Target_Y0, 180, 85)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 350, 85)
        Application.Wait (Now + TimeValue("0:00:10"))
    End If
End If
'select the "R&D Reporting Master Data Set" style
If ResourceStyle Then
    'Style-Button(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 676, 135)
    Application.Wait (Now + TimeValue("0:00:03"))
    'Style-Choice(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 580, ResourceStyleDelta)
    Application.Wait (Now + TimeValue("0:00:15"))
End If
')**Open the OneLinePerResource or Gant Report
'
If VP_iter = 0 Then
    Call z_ExcelSessionWindowMinimized(mywb)
    Call z_ExcelSessionWindowNormal(mywb)
    notused = InputBox("set the style to: BC Report for the Planned Expenditures", , "no entry",
13500, 12500)
    If notused <> "no entry" Then
        Stop
    End If
End If

```

```

'      Application.Wait (Now + TimeValue("0:00:02"))
'      End If
'      Application.Wait (Now + TimeValue("0:00:02"))
'      Call SetForegroundWindow(My_IE_hWnd)
'      End If
'
'      '**Excel Download
'      ***Excel-Download-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1025, 0)
'      Application.Wait (Now + TimeValue("0:00:00"))
'      ***Minimize SmC IE
'      Call ShowWindow(My_IE_hWnd, SW_MINIMIZE)
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:00"))
'      ***make excel active
'      Call z_ExcelSessionWindowNormal(mywb)
'      Call z_ExcelWorkbookWindowNormal(mywb)
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:00"))
'      ***Determine when the Download IE appears
'      'if the time secMax is not exceeded, the download should work out
'      'if the time secMax is exceeded, the download is about to fail, wait an try to close the IE
window
'      'that appears after secMax. After some time this window gets the name HTTP 500 ...
'      strWindowTitle = "Windows Internet Explorer provided by Syngenta"
'      sec = z_IE_WaitUntilNewWindowExists_HTML_Export(strWindowTitle, secMax_Activity)
'      DoEvents
'      ***If IE download fails (HTTP 500 ...), wait and close IE
'      'to be sure SmC is in an idle state before going back to the portfolio module
'      If sec > secMax_Activity Or sec = secMax_Activity Then
'      'wait until SmC is in an idle state
'      Application.Wait (Now + TimeValue("0:02:00")) 'maybe enhance to 10 minutes
'      'try to close the IE window
'      strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided
by Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
'      End If
'      ***minimize IE that prepares the download and wait until the IE download is exported into
Excel
'      strWindowTitle = "Windows Internet Explorer provided by Syngenta"
'      My_IE_XL_hWnd = z_FindWindowHandle(strWindowTitle)
'      If My_IE_XL_hWnd <> 0 Then
'      'minimize IE
'      Call ShowWindow(My_IE_XL_hWnd, SW_MINIMIZE)
'      'give some waiting time until the IE download is exported into Excel
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:10"))
'      End If
'      ***write out the window name
'      WindowName = Workbooks(Workbooks.count).name
'      WorkbooksEntry1 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 5)
'      WorkbooksEntry2 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 6)
'      mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 5) = VP_iter + 1

```

```

' mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 6) = sec
' mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 7) = WindowName
' mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 8) = WorkbooksEntry1
' mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 9) = WorkbooksEntry2
' ***Move the newly created download Wb into the SmC_Download file
' Call z_MoveWbSheetsIntoAnotherWb_V2(mywb, "SmC_A_VP_", VP_iter + 1)
' Application.Wait (Now + TimeValue("0:00:01"))
' ***minimize all Excel workbooks within the Excel session
' Call z_ExcelWorkbookWindowMinimizeAll(mywb)
' DoEvents
' Application.Wait (Now + TimeValue("0:00:01"))
' ***If IE fails, or IE failed to close before, so try to close the window now
' strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet
Explorer provided by Syngenta"
' Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
' Application.Wait (Now + TimeValue("0:00:01"))
' strWindowTitle = "Verify - Windows Internet Explorer provided by Syngenta"
' Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
' Application.Wait (Now + TimeValue("0:00:01"))
' strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided by
Syngenta"
' Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
' Application.Wait (Now + TimeValue("0:00:01"))
' )**Excel Download
' Next VP_iter
' 'save workbook
' mywb.Save
' DoEvents
' Application.Wait (Now + TimeValue("0:00:10"))
' End If
' )**Resource Report
'
' notused = InputBox("Download has been finished", , "no entry", 13500, 12500)
' Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
' Array("Main_DownloadReport", "End", "task: ", CStr(Now()), ""))
'End Sub
'
'
'

```

```

*****
'NOT Replaced by the Word VBA
*****

```

```

"Manual tasks/settings before starting the macro.
' 'Goto:(*Preparations before running the macro
"Tools>References
' 'Visual Basic For Applications
' 'Microsoft Excel 12.0 Object Library
' 'OLE Automation
' 'Microsoft Office 12.0 Object Library
' 'Microsoft Forms 2.0 Object Library

```

```

' 'Microsoft Visual Basic for Applications Extensibility 5.3c:/Program Files (x86)/Common
Files/Microsoft Shared/VBA/VBA6/VBE6EXT.OLB
' 'Microsoft IMAPI2 Base Functionality-c:/Windows/system32/imapi2.dll
' 'Microsoft Internet Controls-c:/Windows/SysWOW64/ieframe.dll
'
'(* Global variables
' '(*Preparations before running the macro
' Private Sh_log_GV As String
' Private delta1_GV As Integer
' Private mywb_GV As Workbook
' '(*Create a new Excel Workbook
' '(*prepare the SmC window in an IE object
' Private My_IE_hWnd_GV As Long
' '(*Open the SmC Project-Module
' Private Target_X0_GV As Long
' Private Target_Y0_GV As Long
' '(*Resource Report
' Private VP_DropdownSecondPos_GV As Long
' '(*PiReport
' Private VP_iter_GV As Integer
' Private TotalVPs_Pi_GV As Integer
' Private PiStyle_GV As Long
' Private PiStyleDelta_GV As Long
' Private SetOnTime_Iter_GV As Integer
'*)* Global variables
'
'(* Constant initialisation
' Public Const SW_RESTORE = 9
' Public Const SW_SHOW = 5
' Public Const SW_MINIMIZE = 6
'*)* Constant initialisation
'
'Private Sub m_Start_Main_DownloadReport_HTMLandXL_Export()
' 'Activity exported with 1997/2000 Export via HTML
' 'PIs exported with 2007/2010 Export via XL (eXcel)
' Dim OnlyPIs As Integer
' Dim myPageURL As String
' Dim answer As String
' *****
' answer = InputBox("stg or pro", , "pro", 13500, 12500)
' If answer = "stg" Then
' myPageURL = "smartchoice.stg.intra"
' ElseIf answer = "pro" Then
' myPageURL = "smartchoice.pro.intra"
' Else
' Stop
' End If
' answer = InputBox("ActivitiesAndPIs = Yes; OnlyPIs = No", , "Yes", 13500, 12500)
' If answer = "Yes" Then
' OnlyPIs = 0 '1=Only PIs, 0=All
' ElseIf answer = "No" Then
' OnlyPIs = 1 '1=Only PIs, 0=All
' Else

```

```

'    Stop
' End If
' Application.Wait (Now + TimeValue("0:00:02"))
' Call m_Main_DownloadReport_HTMLandXL_Export(OnlyPIs, myPageURL)
'End Sub
'
'Private Sub m_Main_DownloadReport_HTMLandXL_Export(OnlyPIs As Integer, myPageURL As
String)
'(*Variable declaration
'  '(*Preparations before running the macro
'  'Dim OnlyPIs As Integer
'  Dim delta2 As Integer
'  Dim WbPath As String
'  Dim wbdate As String
'  Dim WbName As String
'  '(*Create a new Excel Workbook
'  '(*prepare the SmC window in an IE object
'  Dim myIE As SHDocVw.InternetExplorer
'  Dim myPageTitle As String
'  'Dim myPageURL As String
'  Dim strWindowTitle As String 'used several times for different names
'  '(*Open the SmC Project-Module
'  Dim notused As String
'  Dim BySyngentaPortfolio As Boolean
'  '(*ResetButton
'  Dim ResetButtonYes As Integer
'  '(*Resource Report
'  Dim VP_Start As Integer
'  Dim TotalVPs As Integer
'  Dim FailedDownloads_iter As Integer
'  Dim VP_iter As Integer
'  '**Open the OneLinePerResource Report
'  Dim ResourceStyle As Long
'  Dim ResourceStyleDelta As Long
'  '**Workaround for the selection problem: clicks on the scroll bar
'  Dim ScrollClicks_iter As Integer
'  Dim ScrollClicks_iter_Max As Integer
'  '**Excel Download
'  Dim sec As Integer
'  Dim DurationSinceDownloadClick As Integer
'  Dim secMax_Activity As Integer
'  Dim secMax_PI As Integer
'  Dim My_IE_XL_hWnd As Long
'  Dim My_IE_XL_Child_hWnd As Long
'  Dim MyChildName As String
'  Dim WindowName As String
'  Dim WorkbooksEntry1 As String
'  Dim WorkbooksEntry2 As String
'*)*Variable declaration
'
'(*Preparations before running the macro
'  'Manual tasks/settings before starting the macro
'  '0. Reset all global variables

```

```

' Call z_ResetGlobalVariables
' '1. With IE 7 make sure you have only one tab open (the one with SmC)!!!
' '2. Module>Projects Style:Main R&D PI Export_3
' '3. Module>Projects>Open Style: R&D Reporting Master Data Set_5
' '4. Module>Projects>Open Scheduling>Hours&expenditures>One line per resource
' '5. choose only Pi download yes=1 (PiReport) or no=0 (ResourceReport)
' 'default value is 0 (ResourceReport),
' 'if set to 1, then choose the SmC virtual portfolio just before the first PI virtual portfolio
' 'if you set BySyngentaPortfolio=True then you have 15 seconds to set the right VP
' 'OnlyPis = 1 'default=0,
' '6. choose Smc user rights with and without read only
' delta1_GV = 0 'read only: 0, all rights: 17
' delta2 = 0 'read only: 0, all rights: 56
' '7. Calibrate the mouse click on the Open-Module button
' Target_X0_GV = 21 '(calibrate here)
' Target_Y0_GV = 137 '(calibrate here)
' '8. set the path and the name of the workbook that is created (or opened/activated if it already
exists)
' WbPath = "C:\Users\t740698\Desktop\"
' wbdate = z_wbdate(, Now())
' WbName = "CONFIDENTIAL_SmC_Download" & "_" & wbdate & "_V1-0" & ".xlsb"
' 'wbname = "SmC_Download.xlsb"
' '9. Click on the BySyngentaPortfolio-Tab
' 'default value is true (Click)
' 'if set to false, then make sure that the tab in SmC is set to BySyngentaPortfolio
' BySyngentaPortfolio = False 'NoClick=False,Click=True
' '10.Set the Reset-Button for the Resource report
' 'default value is 1
' 'only used in combination with OnlyPis=0 (ResourceReport), no reset with OnlyPis=1 (PiReport)
' ResetButtonYes = 1 'Yes=1, No=0 (does not influence PiReport)
' '11.Resource Report: Number of virtual portfolios
' TotalVPs = 59 '59 if there are 60 (because it starts with 0)
' '12.Resource Style
' 'default=0 (manually set under point 3, no clicking)
' ResourceStyle = 0
' ResourceStyleDelta = 300 'determine the right value
' '13.Max waiting time for the download IE Window
' 'default=35 seconds
' secMax_Activity = 60 '38
' secMax_PI = 220
' '14.After a new SmC release check all positions of buttons that are clicked by the program
' 'find: (new position and mouse click)
' '15.Check from time to time whether the waiting times before new clicks are enough
' '16.Pi Style
' 'default=0 (manually set under point 2, no clicking)
' PiStyle_GV = 0
' PiStyleDelta_GV = 300 'determine the right value
' '17.PiReport: Number of virtual portfolios
' TotalVPs_Pi_GV = 10 '10 if there are 11 Pi virtual portfolios(because it starts with 0)
' '18.IE Zoom must be set to 100% (IE cell in the lower left corner)
' '19.Selection problem workaround
' 'number of clicks on the scroll bar to have all PIs selected
' ScrollClicks_Iter_Max_Big = 9

```

```

' ScrollClicks_Iter_Max_Medium = 7
' ScrollClicks_Iter_Max_Small = 5
' ScrollClicks_Iter_Max_VerySmall = 3
' '20.Check that in the one line per resource report all Attributes are visible
' '21.Set the resource report
' 'none=0, OneLinePerResource=1, Gant=2
' 'default=2 since OneLinePerResource sets back the borderline between table and graph
' ResourceReportFlag = 2
' '22.Make sure Outlook Meeting request reminder does not pop up
'
'*)*Preparations before running the macro
'
'(* Create a new Excel Workbook
' 'show the desktop (minimize all windows)
' Call z_ShowDesktop
' Application.Wait (Now + TimeValue("0:00:01"))
' 'add, open or activate an Excel workbook (and session)
' Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, mywb_GV)
' Application.Wait (Now + TimeValue("0:00:03"))
' 'move and resize excel session
' Call z_ExcelSessionWindowNormal(mywb_GV)
' Call z_ExcelSessionWindowMoveAndResize(mywb_GV, "400", "0", "700", "340")
' Application.Wait (Now + TimeValue("0:00:03"))
' 'minimize all Excel workbooks within the Excel session except mywb_GV
' Call z_ExcelWorkbookWindowMinimizeAll(mywb_GV)
' Call z_ExcelWorkbookWindowNormal(mywb_GV)
' 'do some renaming, deleting and saving
' 'open or activate mywb_GV
' Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, mywb_GV)
' 'bring the Excel session into xlnormal
' Call z_ExcelSessionWindowNormal(mywb_GV)
' Call z_ExcelWorkbookWindowMaximized(mywb_GV)
' 'do the next steps only if they are not already done
' On Error Resume Next
' 'rename logfile
' mywb_GV.Sheets("Sheet1").Select
' Sh_log_GV = "Logfile"
' mywb_GV.Sheets("Sheet1").name = Sh_log_GV
' 'delete empty sheets
' Call z_DeleteWbSheet(mywb_GV, "Sheet2")
' Call z_DeleteWbSheet(mywb_GV, "Sheet3")
' On Error GoTo 0
' 'save workbook
' mywb_GV.Save
'*)* Create a new Excel Workbook
'
'(* prepare the SmC window in an IE object
' 'With IE 7 make sure you have only one tab open (the one with SmC)!!!
' 'or find a solution to toggle between the tabs (send key Ctrl+Tab) until you found the SmC
' 'IE object instantiation
' 'myPageURL = "smartchoice.stg.intra" 'myPageURL = "smartchoice.stg.intra"
' 'Only if not already Open IE: Open IE, Resize, Reposition, Start SmC, Wait 30secs
' myPageTitle = "SmartChoice"

```

```

' Set myIE = IE_Preparation(myPageTitle, myPageURL)
' Application.Wait (Now + TimeValue("0:00:01"))
' 'get the IE handle
' My_IE_hWnd_GV = myIE.hwnd
' 'show or restore IE depending on its current state (iconic = minimized)
' If IsIconic(My_IE_hWnd_GV) Then
'     Call ShowWindow(My_IE_hWnd_GV, SW_RESTORE)
' Else
'     Call ShowWindow(My_IE_hWnd_GV, SW_SHOW)
' End If
' 'bring SmC IE to the foreground
' SetForegroundWindow My_IE_hWnd_GV
")* prepare the SmC window in an IE object
'
' If OnlyPIs = 0 Then
'     Call z_ExcelSessionWindowMinimized(mywb_GV)
'     Call z_ExcelSessionWindowNormal(mywb_GV)
'     notused = InputBox("set the export to 1997-2000", , "no entry", 13500, 12500)
'     If notused <> "no entry" Then
'         Stop
'         Application.Wait (Now + TimeValue("0:00:02"))
'     End If
'     Application.Wait (Now + TimeValue("0:00:02"))
'     Call SetForegroundWindow(My_IE_hWnd_GV)
' End If
'
'(* Open the SmC Project-Module
' Application.Wait (Now + TimeValue("0:00:02"))
' 'Open-Module-Button(new position and mouse click)
' Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 0)
' Application.Wait (Now + TimeValue("0:00:01"))
' 'Project-Module(new position and mouse click)
' Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 57)
' Application.Wait (Now + TimeValue("0:00:15"))
' 'BySyngentaPortfolio-Tab(new position and mouse click)
' If BySyngentaPortfolio Then
'     'new position and mouse click
'     Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 144, 123)
'     Application.Wait (Now + TimeValue("0:00:15"))
' Else
'     Application.Wait (Now + TimeValue("0:00:01"))
' End If
'*)* Open the SmC Project-Module
'
'(*'ResetButton
' If OnlyPIs = 0 Then
'     If ResetButtonYes Then
'         'VirtualPortfolioAdmin-Button(new position and mouse click)
'         Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 238 + delta1_GV, 63)
'         Application.Wait (Now + TimeValue("0:00:01"))
'         'Reset-Button(new position and mouse click)
'         Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 238 + delta1_GV, 148)
'         Application.Wait (Now + TimeValue("0:00:15"))

```



```

' Else
'     Application.Wait (Now + TimeValue("0:00:01"))
' End If
' Else
' 'Select the last activity virtual portfolio, so that the programm starts with
' 'selecting the next virtual portfolio which is the first Pi virtual portfolio
' End If
')*'ResetButton
'
' If OnlyPIs = 0 Then
'     Call z_ExcelSessionWindowMinimized(mywb_GV)
'     Call z_ExcelSessionWindowNormal(mywb_GV)
'     notused = InputBox("set the style to: return to default style on PI report (faster selection)", , "no
entry", 13500, 12500)
'     If notused <> "no entry" Then
'         Stop
'         Application.Wait (Now + TimeValue("0:00:02"))
'     End If
'     Application.Wait (Now + TimeValue("0:00:02"))
'     Call SetForegroundWindow(My_IE_hWnd_GV)
' End If
'
'(* Resource Report
' Application.Wait (Now + TimeValue("0:00:02"))
' If OnlyPIs = 0 Then
'     FailedDownloads_iter = 1
'     VP_Start = 0 '0
'     'loop over the resource VPs
'     For VP_iter = VP_Start To TotalVPs
'
'         '(* Bring IE to the foreground
'         'show or restore IE depending on its current state
'         If IsIconic(My_IE_hWnd_GV) Then
'             Call ShowWindow(My_IE_hWnd_GV, SW_RESTORE)
'         Else
'             Call ShowWindow(My_IE_hWnd_GV, SW_SHOW)
'         End If
'         'bring SmC IE to the foreground
'         Call SetForegroundWindow(My_IE_hWnd_GV)
'         'To be sure SmC is in an idle state before going back to the portfolio module
'         Application.Wait (Now + TimeValue("0:00:03"))
'         ')* Bring IE to the foreground
'
'         '(* close the activities or not and bring SmC IE back to the modul "project"
'         If VP_iter <> 0 Then
'             'Open-Module-Button(new position and mouse click)
'             Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 0)
'             Application.Wait (Now + TimeValue("0:00:03"))
'             'Project-Module(new position and mouse click)
'             Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 57)
'             Application.Wait (Now + TimeValue("0:00:15"))
'         End If
'         ')* close the activities or not and bring SmC IE back to the modul "project"

```

```

'
' (**Choose a new Virtual Portfolio
'   'VirtualPortfolioChoice-DropDown(new position and mouse click)
'   Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 533 + delta1_GV, 63)
'   Application.Wait (Now + TimeValue("0:00:05"))
'   'VirtualPortfolioChoice(new position and mouse click)
'   VP_DropdownSecondPos_GV = 97 'Delta Y0 to click on the second VP
'   Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 338 + delta1_GV,
VP_DropdownSecondPos_GV)
'   Application.Wait (Now + TimeValue("0:00:15"))
'   ')**Choose a new Virtual Portfolio
'
' (**Workaround for the selection problem: clicks on the scroll bar
'   '0 means VP 0+1
'   '1 means VP 1+1
'   '2 means VP 2+1
'   'Nof PIs from 140-200 as well as the last one: VP_Iter = 58 (move it to the right place if it
becomes the second before last)
'   If VP_iter = 19 - 1 Or _
'       VP_iter = 34 - 1 Or _
'       VP_iter = 36 - 1 Or _
'       VP_iter = 39 - 1 Or _
'       VP_iter = 54 - 1 Or _
'       VP_iter = 63 - 1 Then
'       For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Big
'           'Scroll-Bar(new position and mouse click)
'           '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'           Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 1158, 691)
'           Application.Wait (Now + TimeValue("0:00:02"))
'           'SelectAll-Button(new position and mouse click)
'           Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 142)
'           Application.Wait (Now + TimeValue("0:00:01"))
'       Next
'   'Nof PIs from 100-139
'   ElseIf VP_iter = 4 - 1 Or _
'       VP_iter = 18 - 1 Or _
'       VP_iter = 20 - 1 Or _
'       VP_iter = 21 - 1 Or _
'       VP_iter = 37 - 1 Or _
'       VP_iter = 38 - 1 Or _
'       VP_iter = 40 - 1 Or _
'       VP_iter = 42 - 1 Or _
'       VP_iter = 43 - 1 Or _
'       VP_iter = 44 - 1 Or _
'       VP_iter = 45 - 1 Or _
'       VP_iter = 47 - 1 Or _
'       VP_iter = 49 - 1 Or _
'       VP_iter = 53 - 1 Or _
'       VP_iter = 56 - 1 Or _
'       VP_iter = 58 - 1 Or _
'       VP_iter = 61 - 1 Or _
'       VP_iter = 62 - 1 Then

```

```

'      For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Medium
'          'Scroll-Bar(new position and mouse click)
'          '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'          Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 1158, 691)
'          Application.Wait (Now + TimeValue("0:00:02"))
'          'SelectAll-Button(new position and mouse click)
'          Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 142)
'          Application.Wait (Now + TimeValue("0:00:01"))
'      Next
'      'Nof PIs from 60-99
'      Elself VP_iter = 2 - 1 Or _
'          VP_iter = 3 - 1 Or _
'          VP_iter = 6 - 1 Or _
'          VP_iter = 22 - 1 Or _
'          VP_iter = 23 - 1 Or _
'          VP_iter = 29 - 1 Or _
'          VP_iter = 32 - 1 Or _
'          VP_iter = 33 - 1 Or _
'          VP_iter = 41 - 1 Or _
'          VP_iter = 46 - 1 Or _
'          VP_iter = 48 - 1 Or _
'          VP_iter = 50 - 1 Or _
'          VP_iter = 51 - 1 Or _
'          VP_iter = 55 - 1 Or _
'          VP_iter = 59 - 1 Or _
'          VP_iter = 60 - 1 Then
'          For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Small
'              'Scroll-Bar(new position and mouse click)
'              '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'              Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 1158, 691)
'              Application.Wait (Now + TimeValue("0:00:02"))
'              'SelectAll-Button(new position and mouse click)
'              Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 142)
'              Application.Wait (Now + TimeValue("0:00:01"))
'          Next
'          'Nof PIs from 1-59
'          Else
'          For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_VerySmall
'              'Scroll-Bar(new position and mouse click)
'              '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'              Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 1158, 691)
'              Application.Wait (Now + TimeValue("0:00:02"))
'              'SelectAll-Button(new position and mouse click)
'              Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 142)
'              Application.Wait (Now + TimeValue("0:00:01"))
'          Next
'      End If
'      ')*Workaround for the selection problem: clicks on the scroll bar
'
'      '(*Open the OneLinePerResource or Gant Report

```

```

'
'SelectAll-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 142)
Application.Wait (Now + TimeValue("0:00:05"))
'
'Open-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 120 + delta2, 64)
Application.Wait (Now + TimeValue("0:00:03"))
'
'OpenSelectedProjects-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 131 + delta2, 91)
Application.Wait (Now + TimeValue("0:00:15"))
'
'Choose the one line per resource report or the gant report in the first loop
If VP_iter = 0 Then
    '
    If ResourceReportFlag = 1 Then
        '
        'Scheduling-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 29, 30)
        Application.Wait (Now + TimeValue("0:00:03"))
        '
        'Hours&Expenditures-DropDownEntry(new position no click!)
        Call z_SetMousePos(Target_X0_GV, Target_Y0_GV, 29, 163)
        Application.Wait (Now + TimeValue("0:00:03"))
        '
        'OneLinePerResource-DropDownEntry(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 239, 163)
        Application.Wait (Now + TimeValue("0:00:10"))
        '
    ElseIf ResourceReportFlag = 2 Then
        '
        'Scheduling-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 29, 30)
        Application.Wait (Now + TimeValue("0:00:03"))
        '
        'Gant-DropDownEntry(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 29, 63)
        Application.Wait (Now + TimeValue("0:00:10"))
        '
    End If
End If
'
'select the "R&D Reporting Master Data Set" style
If ResourceStyle Then
    '
    'Style-Button(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 676, 135)
    Application.Wait (Now + TimeValue("0:00:03"))
    '
    'Style-Choice(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 580, ResourceStyleDelta)
    Application.Wait (Now + TimeValue("0:00:15"))
    '
End If
')**Open the OneLinePerResource or Gant Report
'
'
If VP_iter = 0 Then
    '
    Call z_ExcelSessionWindowMinimized(mywb_GV)
    Call z_ExcelSessionWindowNormal(mywb_GV)
    '
    notused = InputBox("set the style to: R&D Reporting Master DataSet_6 for the Activity
download", , "no entry", 13500, 12500)
    '
    If notused <> "no entry" Then
        '
        Stop
        '
        Application.Wait (Now + TimeValue("0:00:02"))
        '
    End If
    '
    Application.Wait (Now + TimeValue("0:00:02"))
    '
    Call SetForegroundWindow(My_IE_hWnd_GV)
    '
End If

```

```

'
'
'(**Excel Download
'
'***Excel-Download-Button(new position and mouse click)
'    Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 1025, 0)
'    Application.Wait (Now + TimeValue("0:00:02"))
'
'***Minimize SmC IE
'    Call ShowWindow(My_IE_hWnd_GV, SW_MINIMIZE)
'    DoEvents
'    Application.Wait (Now + TimeValue("0:00:01"))
'
'***make excel active
'    Call z_ExcelSessionWindowNormal(mywb_GV)
'    Call z_ExcelWorkbookWindowNormal(mywb_GV)
'    DoEvents
'    Application.Wait (Now + TimeValue("0:00:01"))
'
'***Determine when the Download IE appears
'    'if the time secMax is not exceeded, the download should work out
'    'if the time secMax is exceeded, the download is about to fail, wait an try to close the IE
window
'    'that appears after secMax. After some time this window gets the name HTTP 500 ...
'    strWindowTitle = "Windows Internet Explorer provided by Syngenta"
'    sec = z_IE_WaitUntilNewWindowExists_HTML_Export(strWindowTitle, secMax_Activity)
'    DoEvents
'
'***If IE download fails (HTTP 500 ...), wait and close IE
'    'to be sure SmC is in an idle state before going back to the portfolio module
'    If sec > secMax_Activity Or sec = secMax_Activity Then
'        'wait until SmC is in an idle state
'        Application.Wait (Now + TimeValue("0:02:00")) 'maybe enhance to 10 minutes
'        'try to close the IE window
'        strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided
by Syngenta"
'        Call z_CloseIE3(strWindowTitle, Sh_log_GV, VP_iter, FailedDownloads_iter)
'        End If
'
'***minimize IE that prepares the download and wait until the IE download is exported into
Excel
'    strWindowTitle = "Windows Internet Explorer provided by Syngenta"
'    My_IE_XL_hWnd = z_FindWindowHandle(strWindowTitle)
'    If My_IE_XL_hWnd <> 0 Then
'        'minimize IE
'        Call ShowWindow(My_IE_XL_hWnd, SW_MINIMIZE)
'        'give some waiting time until the IE download is exported into Excel
'        DoEvents
'        Application.Wait (Now + TimeValue("0:00:10"))
'    End If
'
'***write out the window name
'    WindowName = Workbooks(Workbooks.count).name
'    WorkbooksEntry1 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 5)
'    WorkbooksEntry2 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 6)
'    mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter + 2, 5) = VP_iter + 1
'    mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter + 2, 6) = sec
'    mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter + 2, 7) = WindowName
'    mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter + 2, 8) = WorkbooksEntry1
'    mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter + 2, 9) = WorkbooksEntry2
'
'***Move the newly created download Wb into the SmC_Download file

```

```

'      Call z_MoveWbSheetsIntoAnotherWb_V2(mywb_GV, "SmC_A_VP_", VP_iter + 1)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      ***minimize all Excel workbooks within the Excel session
'      Call z_ExcelWorkbookWindowMinimizeAll(mywb_GV)
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:01"))
'      ***If IE fails, or IE failed to close before, so try to close the window now
'      strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet
Explorer provided by Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log_GV, VP_iter, FailedDownloads_iter)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      strWindowTitle = "Verify - Windows Internet Explorer provided by Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log_GV, VP_iter, FailedDownloads_iter)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided by
Syngenta"
'      Call z_CloseIE3(strWindowTitle, Sh_log_GV, VP_iter, FailedDownloads_iter)
'      Application.Wait (Now + TimeValue("0:00:01"))
'      '**Excel Download
'      Next VP_iter
'      'save workbook
'      mywb_GV.Save
'      DoEvents
'      Application.Wait (Now + TimeValue("0:00:10"))
'      End If
'  )**Resource Report
'
' )(*PiReport
'  '(*close the activities or not and bring SmC IE back to the modul "project"
'  If OnlyPIs = 0 Then
'      'Open-Module-Button(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 0)
'      Application.Wait (Now + TimeValue("0:00:03"))
'      'Project-Module(new position and mouse click)
'      Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 57)
'      Application.Wait (Now + TimeValue("0:00:14"))
'      End If
'  )**close the activities or not and bring SmC IE back to the modul "project"
'
'  If OnlyPIs = 0 Or OnlyPIs = 1 Then
'      Call z_ExcelSessionWindowMinimized(mywb_GV)
'      Call z_ExcelSessionWindowNormal(mywb_GV)
'      notused = InputBox("set the export to 2007-2010 for the PI download", , "no entry", 13500,
12500)
'      If notused <> "no entry" Then
'          Stop
'          Application.Wait (Now + TimeValue("0:00:02"))
'      End If
'      notused = InputBox("set the style to Main R&D PI Export_4 for the PI download", , "no entry",
13500, 12500)
'      If notused <> "no entry" Then
'          Stop
'          Application.Wait (Now + TimeValue("0:00:02"))

```

```

' End If
' notused = InputBox("set the VP before the first PI VP", , "no entry", 13500, 12500)
' If notused <> "no entry" Then
'     Stop
'     Application.Wait (Now + TimeValue("0:00:02"))
' End If
' Application.Wait (Now + TimeValue("0:00:02"))
' Call SetForegroundWindow(My_IE_hWnd_GV)
' End If
'
' (**start the loop
' Application.Wait (Now + TimeValue("0:00:02"))
' If OnlyPIs = 0 Or OnlyPIs = 1 Then
'     'call export PI function which loops over all VPs
'     VP_iter_GV = 0
'     Call z_Export_PiReport_Part1
' End If
' )**start the loop
'(*PiReport
' notused = InputBox("Download has been finished", , "no entry", 13500, 12500)
' Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
'     Array("Main_DownloadReport", "End", "task: ", CStr(Now()), ""))
'End Sub
'
'Private Function z_Export_PiReport_Part1()
'    (**Bring IE to the foreground
'    'show or restore IE depending on its current state
'    If IsIconic(My_IE_hWnd_GV) Then
'        Call ShowWindow(My_IE_hWnd_GV, SW_RESTORE)
'    Else
'        Call ShowWindow(My_IE_hWnd_GV, SW_SHOW)
'    End If
'    'bring SmC IE to the foreground
'    Call SetForegroundWindow(My_IE_hWnd_GV)
'    'To be sure SmC is in an idle state before going back to the portfolio module
'    Application.Wait (Now + TimeValue("0:00:05"))
'    )**Bring IE to the foreground
'
'    (**close the activities or not and bring SmC IE back to the modul "project"
'    If VP_iter_GV = 0 Then
'        'Open-Module-Button(new position and mouse click)
'        Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 0)
'        Application.Wait (Now + TimeValue("0:00:03"))
'        'Project-Module(new position and mouse click)
'        Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 0, 57)
'        Application.Wait (Now + TimeValue("0:00:14"))
'    End If
'    )**close the activities or not and bring SmC IE back to the modul "project"
'
'    (**select the "Main R&D PI Report" style
'    If PiStyle_GV Then
'        'Style-Button(new position and mouse click)
'        Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 1157, 141)

```

```

' Application.Wait (Now + TimeValue("0:00:03"))
' 'Style-Choice(new position and mouse click)
' Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 1096, PiStyleDelta_GV)
' Application.Wait (Now + TimeValue("0:00:10"))
' End If
' '**select the "Main R&D PI Report" style
'
'
' ('**Choose a new Virtual Portfolio
' 'VirtualPortfolioChoice-DropDown(new position and mouse click)
' Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 533 + delta1_GV, 63)
' Application.Wait (Now + TimeValue("0:00:05"))
' 'VirtualPortfolioChoice(new position and mouse click)
' VP_DropdownSecondPos_GV = 97 'Delta Y0 to click on the second VP
' Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 338 + delta1_GV,
VP_DropdownSecondPos_GV)
' Application.Wait (Now + TimeValue("0:00:15"))
' '**Choose a new Virtual Portfolio
'
'
' ('**Excel Download
' '***Excel-Download-Button(new position and mouse click)
' Call z_SetMousePosAndLeftClick(Target_X0_GV, Target_Y0_GV, 1025, 0)
' Application.Wait (Now + TimeValue("0:00:02"))
' '***Minimize SmC IE
' Call ShowWindow(My_IE_hWnd_GV, SW_MINIMIZE)
' DoEvents
' Application.Wait (Now + TimeValue("0:00:01"))
' '***make excel active
' Call z_ExcelSessionWindowNormal(mywb_GV)
' Call z_ExcelWorkbookWindowNormal(mywb_GV)
' DoEvents
' Application.Wait (Now + TimeValue("0:00:01"))
' ('**Excel Download
'
'
' 'call on time to resume with z_SearchWindow
' SetOnTime_Iter_GV = 0
' Call z_SetOnTime("00:00:05", "z_SearchWindow")
'
'
'End Function
'
'Private Function z_SetOnTime(Wait_Sec As Variant, SubName As String)
' 'if something goes wrong stop SetOnTime loop after 60 times
' Static count As Integer
' If count > 60 Then
' Exit Function
' End If
' count = count + 1
' input:Call z_SetOnTime("00:00:05", "z_SearchWindow")
' call SearchWindow with ontime
' Dim TimerSet_i As Variant
' TimerSet_i = Now() + TimeValue(Wait_Sec)
' Application.ontime EarliestTime:=TimerSet_i, Procedure:=SubName, Schedule:=True
' 'now VBA is not running and waiting for the download window to pop up
'End Function

```



```

'
'Private Sub z_SearchWindow()
' 'search the window
' Dim strWindowTitle As String
' Dim MyAppHWND As Long
' strWindowTitle = "Microsoft Excel - export-t740698-*"
' MyAppHWND = z_FindWindowLike(strWindowTitle)
' 'if found: call download_PI
' If MyAppHWND <> 0 Then
'     Call z_Export_PiReport_Part2
' 'if not found: call ontime
' Else
'     SetOnTime_Iter_GV = SetOnTime_Iter_GV + 1
'     Call z_SetOnTime("00:00:05", "z_SearchWindow")
' End If
" 'search the window
" If Workbooks(Workbooks.Count).Worksheets(1).Name = "Projects" Then
"     Workbooks(Workbooks.Count).Worksheets(1).Name = "Projects_"
"     Call z_Export_PiReport_Part2
" Else
"     SetOnTime_Iter_GV = SetOnTime_Iter_GV + 1
"     Call z_SetOnTime("00:00:05", "z_SearchWindow")
" End If
'End Sub
'Private Sub z_SearchWindow2()
' 'function not used
'
" 'search the window
" Dim strWindowTitle As String
" Dim MyAppHWND As Long
" strWindowTitle = "Microsoft Excel - export-t740698-*"
" MyAppHWND = z_FindWindowLike(strWindowTitle)
" 'if found: call download_PI
" If MyAppHWND <> 0 Then
"     Call z_Export_PiReport_Part2
" 'if not found: call ontime
" Else
"     SetOnTime_Iter_GV = SetOnTime_Iter_GV + 1
"     Call z_SetOnTime("00:00:05", "z_SearchWindow")
" End If
' 'search the window
' If Workbooks(Workbooks.count).Worksheets(1).name = "Projects" Then
'     Workbooks(Workbooks.count).Worksheets(1).name = "Projects_"
'     Call z_Export_PiReport_Part2
' Else
'     SetOnTime_Iter_GV = SetOnTime_Iter_GV + 1
'     Call z_SetOnTime("00:00:05", "z_SearchWindow")
' End If
'End Sub
'Private Sub z_Export_PiReport_Part2()
' '(*move the export into the download workbook
'     ***write out the window name
'     Dim WindowName As String

```

```

' Dim WorkbooksEntry1 As String
' Dim WorkbooksEntry2 As String
' Dim sec As String
' sec = (SetOnTime_Iter_GV + 1) * 5
' WindowName = Workbooks(Workbooks.count).name
' WorkbooksEntry1 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 5)
' WorkbooksEntry2 = Workbooks(Workbooks.count).Sheets(1).Cells(2, 6)
' mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter_GV + 2, 5) = VP_iter_GV + 1
' mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter_GV + 2, 6) = sec
' mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter_GV + 2, 7) = WindowName
' mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter_GV + 2, 8) = WorkbooksEntry1
' mywb_GV.Worksheets(Sh_log_GV).Cells(VP_iter_GV + 2, 9) = WorkbooksEntry2
' ***Save the export workbook
' Application.DisplayAlerts = False
' Call z_WorkbookSave(ActiveWorkbook)
' Application.DisplayAlerts = True
' ***Move newly created download Wb into the SmC_Download file
' Call z_MoveWbSheetsIntoAnotherWb_V2(mywb_GV, "SmC_P_VP_", VP_iter_GV + 1)
' Application.Wait (Now + TimeValue("0:00:01"))
' ***Save the download workbook
' Call z_WorkbookSave(mywb_GV)
' )**move the export into the download workbook
'
' 'iterate the global variable VP_Iter_GV
' VP_iter_GV = VP_iter_GV + 1
' If VP_iter_GV < TotalVPs_Pi_GV Or VP_iter_GV = TotalVPs_Pi_GV Then
'     'next export
'     Call z_Export_PiReport_Part1
' Else
'     'download complete
' End If
' End Sub
'End Sub

'Private Function z_ResetGlobalVariables()
'(* Global variables
' (*Preparations before running the macro
' Sh_log_GV = ""
' delta1_GV = 0
' Set mywb_GV = Nothing
' (*Create a new Excel Workbook
' (*prepare the SmC window in an IE object
' My_IE_hWnd_GV = 0
' (*Open the SmC Project-Module
' Target_X0_GV = 0
' Target_Y0_GV = 0
' (*Resource Report
' VP_DropdownSecondPos_GV = 0
' (*PiReport
' VP_iter_GV = 0
' TotalVPs_Pi_GV = 0
' PiStyle_GV = 0
' PiStyleDelta_GV = 0
' SetOnTime_Iter_GV = 0

```

```

")*Global variables
'End Function
'

```

```

Sub m_Start_Main_GenerateReport()
    Dim flag As Integer
    Dim PI_Export As String
    '1 to run the report, 0 to reopen and/or rerun a report
    flag = 1
    *****

    PI_Export = "2010" & ""1997"
    *****

    Call m_Main_GenerateReport(flag, PI_Export)
End Sub

```

```

Sub m_ReStart_Main_GenerateReport()
    Dim flag As Integer
    Dim PI_Export As String
    Dim flag_rerun_Generation_Part1To4 As Integer
    Dim flag_rerun_Generation_Part2To4 As Integer
    Dim flag_rerun_Generation_Part3To4 As Integer
    Dim flag_rerun_Generation_Part4To4 As Integer
    Dim wbdate As String
    '1 to run the report, 0 to reopen and/or rerun a report
    flag = 0
    *****

    PI_Export = "2010" & ""1997"
    wbdate = "2012_10_16"

    'Only used if flag=0
    '1: to rerun from the code from Creating Sheet Generation_Part1.
    flag_rerun_Generation_Part1To4 = 0
    '1: to rerun from the code from Creating Sheet Generation_Part2.
    flag_rerun_Generation_Part2To4 = 0
    '1: to rerun from the code from Creating Sheet Generation_Part3.
    flag_rerun_Generation_Part3To4 = 0
    '1: to rerun from the code from Creating Sheet Generation_Part4.
    flag_rerun_Generation_Part4To4 = 1
    *****

    Call m_Main_GenerateReport(flag, PI_Export, wbdate, _
        flag_rerun_Generation_Part1To4, flag_rerun_Generation_Part2To4,
flag_rerun_Generation_Part3To4, flag_rerun_Generation_Part4To4)
End Sub

```

```

Sub m_Main_GenerateReport(flag As Integer, PI_Export As String, Optional wbdate As String, _
    Optional flag_rerun_Generation_Part1To4 As Integer, _
    Optional flag_rerun_Generation_Part2To4 As Integer, _
    Optional flag_rerun_Generation_Part3To4 As Integer, _
    Optional flag_rerun_Generation_Part4To4 As Integer)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task:", "file:", CStr(Now()), ""))
    *****

```

```

'Dim flag As Integer
Dim flag_reopen_wb As Integer
'Dim flag_rerun_Generation_Part1To3 As Integer
'Dim flag_rerun_Generation_Part2To3 As Integer
'Dim wbdate As String
Dim downloadversion As String
Dim reportversion As String
'Dim PI_Export As String
*****

'input
downloadversion = "_V1-0" 'output of the a01_Main_DownloadReport
'flag = 1 '1 to run the report, 0 to reopen and/or rerun a report
'PI_Export = "2010" ""1997"
If flag Then
    'run
    wbdate = z_wbdate(, Now())
    reportversion = "_V1-0" 'output of a02_Main_GenerateReport
    folderdescription = ""
Else
    'reopen
    flag_reopen_wb = 1 '1 to open/close and set existing wb's. Only used if flag=0
    'flag_rerun_Generation_Part1To4 = 0 'to rerun from the code from Creating Sheet
    Generation_Part1.Only used if flag=0
    'flag_rerun_Generation_Part2To4 = 1 'to rerun from the code from Creating Sheet
    Generation_Part2.Only used if flag=0
    reportversion = "_V1-0"
    folderdescription = ""
End If
*****

'Open and activate an Excel workbook (and session)
Dim Wb_Download As Workbook
Dim WbPath As String
Dim WbName As String
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
    folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_Download" & "_" & wbdate & downloadversion & ".xlsb"
Else
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
    folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_Download" & "_" & wbdate & downloadversion & ".xlsb"
    If flag_rerun_Generation_Part1To4 = 1 Then 'If flag_reopen_wb Then
        On Error Resume Next
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_Download)
        On Error GoTo 0
    End If
End If
If flag Then
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_Download)
End If
If flag Then
    *****

    'move and resize the Excel session

```

```

Call z_ExcelSessionWindowNormal(Wb_Download)
Call z_ExcelSessionWindowMoveAndResize(Wb_Download, "0", "0", "700", "600")
End If
*****
*****

'Produce the numbers to make tests on the input (SmC download)
Dim Wb_ExpNofVPs As Workbook
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
WbName = "Expected_Nof_VirtualPortfolios.xlsb"
*****
*****

'open the VP check
If flag Then
    Call z_ShNew("Log_0", "Begin")
End If
If flag Then
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_ExpNofVPs)
End If
'move the VP check into Wb_Download
If flag Then
    Call z_CopySheetFromWb1ToWb2("ExpectedNofVPs", Wb_ExpNofVPs, Wb_Download, "Begin")
End If
If flag Then
    'close Wb_VPCheck, closes the active workbook and saves any changes
    Wb_ExpNofVPs.Close True
End If
If flag Then
    MsgBox ("Start the renaming of the sheets according to sheet ExpectedNofVPs")
    Stop
    Call z_CorrectTheSheetNames("ExpectedNofVPs", Wb_Download)
    Stop
End If
*****

'flattening of the Pi level worksheets Smc_P
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Flattening PI downloads", "file: All SmC_P ->
SmC_P..._c", CStr(Now()), ""))
    Dim Sh_from As Worksheet
    Dim Sh_to As String
    'loop over all worksheets
    For Each Sh_from In Wb_Download.Sheets
        If Sh_from.name <> "PMEC_VBA_MasterDataSet_1" And Sh_from.name <> "Log_1" And
Sh_from.name <> "Logfile" Then
            If left(Sh_from.name, 5) = "SmC_P" Then
                Sh_to = Sh_from.name & "_c"
                If PI_Export = "2010" Then
                    Dim Sh_New As String
                    Sh_New = Sh_from.name & "_tmp"
                    Call z_CopySheet2(Sh_from.name, Sh_New, "Begin")
                    Call z_SmC_CleanProjectReportPortfolioStructure_2010Export(Sh_New, Sh_to)

```

```

        Elself PI_Export = "1997" Then
            Call z_SmC_CleanProjectReportPortfolioStructure_1997Export(Sh_from.name, Sh_to)
        Else
            Stop
        End If
    End If
End If
Next Sh_from
Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
    Array("Main_CreateReport", "End", "task: Flattening PI downloads", "file: All SmC_P ->
SmC_P..._c", CStr(Now()), ""))
End If
*****

If flag Then
    *****

    Call z_CreateSh_VPCheck("ExpectedNofVPs", "Log_0")
    *****

End If
If flag Then
    'in Log_0 add Column Names
    Call z_AddColNames(Array("Sheet Name", "Nof Pis Actual", "Nof Pis Plan", "Difference",
"Remarks"), "Log_0", 1, 2)
    'map numbers from VP_Check into Log_0
    'do not use the FastVersion (because of the sort function included)!
    Call z_ShMapColumns("ExpectedNofVPs", "Sheet Name", Array("Nof Pis"), "Log_0", "Sheet Name",
Array("Nof Pis Plan"))
End If
If flag Then
    'Add the differences
    Call z_Insert_DifferenceFormula("C", "D", 5, "Sheet Name", 1, "Log_0")
End If
    'Copy the sheet into the summary workbook
    Dim Wb_InputChkOnVP As Workbook
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Outputs_MainGenerateReport\"
    WbName = "InputCheckOn_VirtualPortfolios.xlsb"
    *****
    *****

If flag Then
    MsgBox ("Check the nof VPs on the sheet Log_0")
    Stop
End If
    'open
If flag Then
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_InputChkOnVP)
End If
    'move
If flag Then
    Call z_CopySheetFromWb1ToWb2("Log_0", Wb_Download, Wb_InputChkOnVP, "Begin")
End If
    'rename
If flag Then
    Wb_InputChkOnVP.Sheets("Log_0").name = wbdate

```

```

End If
If flag Then
    'close Wb_RBS, closes the active workbook and saves any changes
    Wb_InputChkOnVP.Close True
End If
'*****

'loop over all worksheets and copy the content of the activity level SmC_A sheets
'into the MasterDataSet
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Copy/Paste SmC_A sheets into", "file:
MasterDataSheet_1", CStr(Now()), ""))
    Call z_ShNew("Log_1", "Begin")
    Call z_ShNew("PMEC_VBA_MasterDataSet_1", "Begin")
    Call z_ShNew("PMEC_VBA_PiDataSet_1", "Begin")
    For Each Sh_from In Wb_Download.Sheets
        'write the data from Sh_from into the MasterDataSet
        If Sh_from.name <> "PMEC_VBA_MasterDataSet_1" And Sh_from.name <> "Log_1" And
Sh_from.name <> "Logfile" Then
            If Left(Sh_from.name, 5) = "SmC_A" Then
                Call z_CopySh1ToSh2_GivenColLastRow(Sh_from.name, "Activity Identifier",
"PMEC_VBA_MasterDataSet_1")
            End If
        End If
    Next Sh_from
End If
'loop over all worksheets and copy the content of the Pi level SmC_P sheets
'into the PiDataSet
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Copy/Paste SmC_P sheets into", "file:
PMEC_VBA_PiDataSet_1", CStr(Now()), ""))
    For Each Sh_from In Wb_Download.Sheets
        'write the data from Sh_from into the MasterDataSet
        If Sh_from.name <> "PMEC_VBA_MasterDataSet_1" And Sh_from.name <> "Log_1" And
Sh_from.name <> "Logfile" Then
            If Right(Sh_from.name, 2) = "_c" Then
                Call z_CopySh1ToSh2_GivenColLastRow(Sh_from.name, "Portfolio Level 1",
"PMEC_VBA_PiDataSet_1")
            End If
        End If
    Next Sh_from
End If
'*****
'*****

'Adds the new workbook RD_MasterDataSet
'*****
'*****

'Run the code from here
If flag = 0 Then
    If flag_rerun_Generation_Part1To4 = 1 Then
        flag = 1
    End If

```

```

End If
'Add a new workbook and move the MasterDataSet and the PiDataSet into it
Dim Wb_GenPart1 As Workbook
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part1_" & wbdate &
reportversion & ".xlsb"
Else
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part1_" & wbdate &
reportversion & ".xlsb"
    If flag_rerun_Generation_Part2To4 = 1 Then 'If flag_reopen_wb Then
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart1)
    End If
End If
If flag Then
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, Wb_GenPart1)
End If
If flag Then
    Call z_CopySheetFromWb1ToWb2("Log_1", Wb_Download, Wb_GenPart1, "Begin")
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_PiDataSet_1", Wb_Download, Wb_GenPart1,
"Begin")
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_1", Wb_Download, Wb_GenPart1,
"Begin")
    'in case they have already been deleted in a previous run
    On Error Resume Next
    Call z_ShDelete("Sheet1", Wb_GenPart1)
    Call z_ShDelete("Sheet2", Wb_GenPart1)
    Call z_ShDelete("Sheet3", Wb_GenPart1)
    On Error GoTo 0
    Call z_WorkbookSave(Wb_GenPart1)
End If
If flag Then
    'close Wb_Download, closes the active workbook and saves any changes
    Wb_Download.Close True
Else
    'Wb_Download.Close False
End If
'*****

'move and resize the Excel session
If flag Then
    Call z_ExcelSessionWindowNormal(Wb_GenPart1)
    Call z_ExcelSessionWindowMoveAndResize(Wb_GenPart1, "0", "0", "900", "700")
End If
'*****

'Processing of the RD_MasterDataSet
'*****

'*****

'remove the first empty row and the column names in between the PiDataSet
If flag Then

```



```

Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
    Array("Main_CreateReport", "Begin", "task:Remove rows and cols", "file:
PMEC_VBA_PiDataSet_2", CStr(Now()), ""))
Call z_ShNew("Log_2", "Begin")
Call z_ShNewFlatValueCopy("PMEC_VBA_PiDataSet_1", "PMEC_VBA_PiDataSet_2", "Begin")
Call z_DeleteRows("PMEC_VBA_PiDataSet_2", 1, 1)
Call z_RemoveColNames("PMEC_VBA_PiDataSet_2", "Log_2", 1, "Portfolio Level 1")
End If
*****

'Write to each row of the MasterDataSet the Ids (Pild, Wsld, Tkld, Actld, Resld, Sortld)
'1.add col A, col B, col C and col D
'2.go through Col_from (Pild, Actld, Resld)
'2.1(if (left(Col_from,2))= PI then) write Pild into col A
'2.2 col B
'2.3 col C
'2.4(if (left(Col_from,2))= PI,WS,TK,MS) write Actld into col D
'2.5(if (left(Col_from,2))= empty) write MyResld into col E
'2.6(if (left(Col_from,2))= PI then) write MySortld=1..PiRows.Count into col F
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task:Add and Fill 6 ID cols", "file:
PMEC_VBA_MasterDataSet_2", CStr(Now()), ""))
    Call z_ShNew("Log_3", "Begin")
    Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_1", "PMEC_VBA_MasterDataSet_2",
"Begin")
    Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_2", 6, 1)
    Call z_DeleteRows("PMEC_VBA_MasterDataSet_2", 1, 1)
    Call z_AddColNames(Array("PildIdentifier", "WsIdentifier", "TkIdentifier", "ActivityIdentifier",
"MyResourceId", "MySortId"), "PMEC_VBA_MasterDataSet_2", 1)
    Dim Col_First As Long
    Col_First = z_GetColumnIndex("Activity Identifier", 1, "PMEC_VBA_MasterDataSet_2")
    Call z_RemoveColNames("PMEC_VBA_MasterDataSet_2", "Log_3", Col_First, "Activity Identifier")
    Call z_AddWBSPild("PMEC_VBA_MasterDataSet_2", "Activity Identifier", "PildIdentifier")
    Call z_AddWBSWsld("PMEC_VBA_MasterDataSet_2", "Activity Identifier", "WsIdentifier")
    Call z_AddWBSTkld("PMEC_VBA_MasterDataSet_2", "Activity Identifier", "TkIdentifier")
    Call z_AddWBSActivityld("PMEC_VBA_MasterDataSet_2", "Activity Identifier", "ActivityIdentifier")
    Call z_AddMyResourceId("PMEC_VBA_MasterDataSet_2", "Activity Identifier", "MyResourceId")
    Call z_AddMySortId("PMEC_VBA_MasterDataSet_2", "ActivityIdentifier", "MySortId",
"WithoutMS")
    Call z_WorkbookSave(Wb_GenPart1)
End If
*****

'Copy the values of the summary lines into the lines below
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Copy 14 attributes from Activity into Resource
lines", "file: PMEC_VBA_MasterDataSet_3", CStr(Now()), ""))
    Call z_ShNew("Log_4", "Begin")
    Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_2", "PMEC_VBA_MasterDataSet_3",
"Begin")
    Dim ColName_Array() As Variant
    Dim ColName As String
    Dim col As Long

```

```

' ColName_Array = Array("Task Title", "Description", "Activity Comment", "Activity type", _
'     "Task Customer", "Task Contact", "Task Location", _
'     "Task Status", "Duration", "Planned start", "Expected finish export", "Actual start", "Actual
Finish export") ""Planned start"

Dim Wb_from As Workbook
Dim Rng As Range
Set Rng = z_RangeOfWb_AttributeNamesFromSheetToArray("GenerateReport", 2, Wb_from)
ColName_Array = z_ReadAttributeNames_FromSheet_ToArray("GenerateReport", Rng,
"ReadACol", Wb_from)

For iter = LBound(ColName_Array) To UBound(ColName_Array)
    ColName = ColName_Array(iter)
    Call z_CopyWBSAttributeEntriesIntoResources_overwrite("PMEC_VBA_MasterDataSet_3",
"ActivityIdentifier", ColName)
Next iter
'Remark: SyngentaPortfolio=PI Syngenta Portfolio, Syngenta Portfolio=Activity Syngenta Portfolio
ColName = "Syngenta Portfolio"
Call z_CopyWBSAttributeEntriesIntoResources_not_overwrite("PMEC_VBA_MasterDataSet_3",
"PIIdentifier", ColName)
'Call z_CopyWBSAttributeEntriesIntoResources_overwrite("PMEC_VBA_MasterDataSet_3",
"PIIdentifier", ColName)
Call z_WorkbookSave(Wb_GenPart1)
End If
*****

'Add empty columns and write into the first row the column names
Dim FirstYr As Integer
FirstYr = VBA.DateTime.year(Now()) - 1
Dim Col_to As Long
'Full costs
If flag Then
    Call z_ShNew("Log_5", "Begin")
    Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_3", "PMEC_VBA_MasterDataSet_4",
"Begin")
    'add EAC Full Cost columns
    Col_to = z_GetColumnIndex("ETC Full Costs", 1, "PMEC_VBA_MasterDataSet_4") + 1
    Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
    Call z_AddColNames(Array("EAC Full Costs " & CStr(FirstYr) & "_c", "EAC Full Costs " & CStr(FirstYr +
1) & "_c", _
        "EAC Full Costs " & CStr(FirstYr + 2) & "_c", "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
        "EAC Full Costs " & CStr(FirstYr + 4) & "_c", "EAC Full Costs" & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
    'add EAC SD Full Costs columns
    Col_to = z_GetColumnIndex("EAC Full Costs" & "_c", 1, "PMEC_VBA_MasterDataSet_4") + 1
    Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
    Call z_AddColNames(Array("EAC SD Full Costs " & CStr(FirstYr) & "_c", "EAC SD Full Costs " &
CStr(FirstYr + 1) & "_c", _
        "EAC SD Full Costs " & CStr(FirstYr + 2) & "_c", "EAC SD Full Costs " & CStr(FirstYr + 3) & "_c",
_
        "EAC SD Full Costs " & CStr(FirstYr + 4) & "_c", "EAC SD Full Costs" & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
    'add EAC Trials Full Cost columns
    Col_to = z_GetColumnIndex("EAC SD Full Costs" & "_c", 1, "PMEC_VBA_MasterDataSet_4") + 1

```

```

Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Trials Full Costs " & CStr(FirstYr) & "_c", "EAC Trials Full Costs " &
CStr(FirstYr + 1) & "_c", _
    "EAC Trials Full Costs " & CStr(FirstYr + 2) & "_c", "EAC Trials Full Costs " & CStr(FirstYr + 3) &
"_c", _
    "EAC Trials Full Costs " & CStr(FirstYr + 4) & "_c", "EAC Trials Full Costs" & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
'add EACOther$ columns
Col_to = z_GetColumnIndex("EAC Trials Full Costs" & "_c", 1, "PMEC_VBA_MasterDataSet_4") + 1
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Other $ " & CStr(FirstYr) & "_c", "EAC Other $ " & CStr(FirstYr + 1)
& "_c", _
    "EAC Other $ " & CStr(FirstYr + 2) & "_c", "EAC Other $ " & CStr(FirstYr + 3) & "_c", _
    "EAC Other $ " & CStr(FirstYr + 4) & "_c", "EAC Other $" & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
'add EACExt$ columns
Col_to = z_GetColumnIndex("EAC Other $" & "_c", 1, "PMEC_VBA_MasterDataSet_4") + 1
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Ext $ " & CStr(FirstYr) & "_c", "EAC Ext $ " & CStr(FirstYr + 1) &
"_c", _
    "EAC Ext $ " & CStr(FirstYr + 2) & "_c", "EAC Ext $ " & CStr(FirstYr + 3) & "_c", _
    "EAC Ext $ " & CStr(FirstYr + 4) & "_c", "EAC Ext $" & "_c"), "PMEC_VBA_MasterDataSet_4",
1, Col_to)
End If
'Direct costs
If flag Then
'add EAC Direct Cost columns
Col_to = z_GetColumnIndex("ETC Direct Costs", 1, "PMEC_VBA_MasterDataSet_4") + 1
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Direct Costs " & CStr(FirstYr) & "_c", "EAC Direct Costs " &
CStr(FirstYr + 1) & "_c", _
    "EAC Direct Costs " & CStr(FirstYr + 2) & "_c", "EAC Direct Costs " & CStr(FirstYr + 3) & "_c", _
    "EAC Direct Costs " & CStr(FirstYr + 4) & "_c", "EAC Direct Costs" & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
'add EAC SD Direct Costs columns
Col_to = z_GetColumnIndex("EAC Direct Costs" & "_c", 1, "PMEC_VBA_MasterDataSet_4") + 1
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC SD Direct Costs " & CStr(FirstYr) & "_c", "EAC SD Direct Costs " &
CStr(FirstYr + 1) & "_c", _
    "EAC SD Direct Costs " & CStr(FirstYr + 2) & "_c", "EAC SD Direct Costs " & CStr(FirstYr + 3) &
"_c", _
    "EAC SD Direct Costs " & CStr(FirstYr + 4) & "_c", "EAC SD Direct Costs" & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
'add EAC Trials Direct Cost columns
Col_to = z_GetColumnIndex("EAC SD Direct Costs" & "_c", 1, "PMEC_VBA_MasterDataSet_4") + 1
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Trials Direct Costs " & CStr(FirstYr) & "_c", "EAC Trials Direct Costs
" & CStr(FirstYr + 1) & "_c", _
    "EAC Trials Direct Costs " & CStr(FirstYr + 2) & "_c", "EAC Trials Direct Costs " & CStr(FirstYr +
3) & "_c", _
    "EAC Trials Direct Costs " & CStr(FirstYr + 4) & "_c", "EAC Trials Direct Costs" & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
'add EACOther$ columns

```

```

Col_to = z_GetColumnIndex("EAC Trials Direct Costs" & "_c", 1, "PMEC_VBA_MasterDataSet_4") +
1
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Other $ Direct Costs " & CStr(FirstYr) & "_c", "EAC Other $ Direct
Costs " & CStr(FirstYr + 1) & "_c", _
    "EAC Other $ Direct Costs " & CStr(FirstYr + 2) & "_c", "EAC Other $ Direct Costs " &
CStr(FirstYr + 3) & "_c", _
    "EAC Other $ Direct Costs " & CStr(FirstYr + 4) & "_c", "EAC Other $ Direct Costs " & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
'add EACExt$ columns
Col_to = z_GetColumnIndex("EAC Other $ Direct Costs" & "_c", 1, "PMEC_VBA_MasterDataSet_4")
+ 1
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Ext $ Direct Costs " & CStr(FirstYr) & "_c", "EAC Ext $ Direct Costs "
& CStr(FirstYr + 1) & "_c", _
    "EAC Ext $ Direct Costs " & CStr(FirstYr + 2) & "_c", "EAC Ext $ Direct Costs " & CStr(FirstYr +
3) & "_c", _
    "EAC Ext $ Direct Costs " & CStr(FirstYr + 4) & "_c", "EAC Ext $ Direct Costs " & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
End If
'Number of SD and Trials
If flag Then
'add EAC # SD Direct Costs columns
Col_to = z_GetColumnIndex("ETC # SD", 1, "PMEC_VBA_MasterDataSet_4") + 1
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC # SD " & CStr(FirstYr) & "_c", "EAC # SD " & CStr(FirstYr + 1) &
"_c", _
    "EAC # SD " & CStr(FirstYr + 2) & "_c", "EAC # SD " & CStr(FirstYr + 3) & "_c", _
    "EAC # SD " & CStr(FirstYr + 4) & "_c", "EAC # SD" & "_c"), "PMEC_VBA_MasterDataSet_4",
1, Col_to)
'add EAC # Trials Direct Cost columns
Col_to = z_GetColumnIndex("EAC # SD" & "_c", 1, "PMEC_VBA_MasterDataSet_4") + 1
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC # Trials " & CStr(FirstYr) & "_c", "EAC # Trials " & CStr(FirstYr + 1)
& "_c", _
    "EAC # Trials " & CStr(FirstYr + 2) & "_c", "EAC # Trials " & CStr(FirstYr + 3) & "_c", _
    "EAC # Trials " & CStr(FirstYr + 4) & "_c", "EAC # Trials" & "_c"),
"PMEC_VBA_MasterDataSet_4", 1, Col_to)
End If

'*****

'sort the dataset using col 3 to keep the resources (or col 4 to only remove the MS but keep the
summary lines)
If flag Then
Call z_ShNew("Log_6_2", "Begin")
Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_4", "PMEC_VBA_MasterDataSet_5",
"Begin")
Col_to = z_GetColumnIndex("MyResourceId", 1, "PMEC_VBA_MasterDataSet_5")
Call z_Sort("PMEC_VBA_MasterDataSet_5", Col_to, Col_to)
Call z_DeleteNotResourceRows("PMEC_VBA_MasterDataSet_5", "MyResourceId")
Call z_WorkbookSave(Wb_GenPart1)
End If
If flag Then

```

```

'=====
'=====
'-open new process
'-open this workbook in the new process (read only)
'-set on time to run this workbook in the new process
'-close this workbook (read/write)
'-close this process
'-wait until on time starts the procedure resume
'-resume sets this workbook in the new process to read/write
'-resume calls the sub m_ReStart_Main_GenerateReport
'-the flag_rerun_Generation_Part2To4 set to 1 resumes the code here
Call m_Main_MemoryLeakWorkaround_1
'=====
'=====
End If
'*****
'*****

'Adds a new workbook RD_MasterDataSet because of memory problems
'*****
'*****

'Rerun the code from here
If flag = 0 Then
    If flag_rerun_Generation_Part2To4 = 1 Then
        flag = 1
    End If
End If
'Add a new workbook and move the MasterDataSet and the PiDataSet into it
Dim Wb_GenPart2 As Workbook
If flag Then
Else
    If flag_rerun_Generation_Part2To4 = 1 Then 'If flag_reopen_wb Then
        Wb_GenPart1.Close False
    End If
End If
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part2_" & wbdate &
reportversion & ".xlsb"
Else
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part2_" & wbdate &
reportversion & ".xlsb"
    If flag_rerun_Generation_Part3To4 = 1 Then 'If flag_reopen_wb Then
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart2)
    End If
End If
If flag Then
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, Wb_GenPart2)
End If
If flag Then
    'copy the last datasheet into the new workbook

```

```

On Error Resume Next
Sheets("PMEC_VBA_MasterDataSet_5").Activate
If Err.Number <> 0 Then
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_5", Wb_GenPart1,
Wb_GenPart2, "Begin")
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_PiDataSet_2", Wb_GenPart1, Wb_GenPart2,
"Begin")
End If
On Error GoTo 0
'in case they have already been deleted in a previous run
On Error Resume Next
    Call z_ShDelete("Sheet1", Wb_GenPart2)
    Call z_ShDelete("Sheet2", Wb_GenPart2)
    Call z_ShDelete("Sheet3", Wb_GenPart2)
On Error GoTo 0
Call z_WorkbookSave(Wb_GenPart2)
End If
If flag Then
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_GenPart1.Close True
End If
*****

'change the format and start the cost calculation
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Calculate 72 Cost Cols", "file:
PMEC_VBA_MasterDataSet_6", CStr(Now()), ""))
    Call z_ShNew("Log_7-1", "Begin")
    Call z_ShNew("Log_7-2", "Begin")
    Call z_ShNew("Log_7-3", "Begin")
    Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_5", "PMEC_VBA_MasterDataSet_6",
"Begin")
    Dim FromCol As Long
    Dim ToCol As Long
    'FromCol = first cost column
    FromCol = z_GetColumnIndex("AC # Trials " & CStr(FirstYr), 1, "PMEC_VBA_MasterDataSet_6")
    'ToCol = last cost column
    ToCol = z_GetColumnIndex("EAC Ext $ Direct Costs" & "_c", 1, "PMEC_VBA_MasterDataSet_6")
    Call z_ChgFmt_CostCols("PMEC_VBA_MasterDataSet_6", FromCol, ToCol)

    'Input the AC and ETC numbers, calculate and output the EAC numbers
    'Full Costs
    Call FullCostCalc("PMEC_VBA_MasterDataSet_6", "PiIdentifier", FirstYr, "AC Full Costs", "ETC Full
Costs", "Unit", _
        "EAC SD Full Costs", "EAC Trials Full Costs", "EAC Other $", "EAC Ext $", "EAC Full Costs",
"Log_7-1")
    'Direct Costs
    Call DirectCostCalc("PMEC_VBA_MasterDataSet_6", "PiIdentifier", FirstYr, "AC Direct Costs", "ETC
Direct Costs", "Unit", _
        "EAC SD Direct Costs", "EAC Trials Direct Costs", "EAC Other $ Direct Costs", "EAC Ext $
Direct Costs", _
        "EAC Direct Costs", "Log_7-2")
    'Number of Trials and SD

```

```

'MsgBox ("Number calc einbauen ---- testen")
'Stop
Call NumbersCalc("PMEC_VBA_MasterDataSet_6", "PIdentifier", FirstYr, _
    "AC # Trials", "ETC # Trials", "AC # SD", "ETC # SD", "Unit", _
    "EAC # SD", "EAC # Trials", "Log_7-3") '!!!!!!!Testen

    Call z_WorkbookSave(Wb_GenPart2)
End If
If flag Then
    'remove the resources with EAC Full Costs==0
    'MsgBox "check the nof rows and the nof rows with EAC Full Costs_c == 0"
    'Stop
    Dim Col_EacFullCosts As Long
    Col_EacFullCosts = z_GetColumnIndex("EAC Full Costs_c", 1, "PMEC_VBA_MasterDataSet_6")
    Call z_DeleteRowsOfEmptyCells("PMEC_VBA_MasterDataSet_6", Col_EacFullCosts, 2)
    'MsgBox "check the nof rows after"
    'Stop
    Call z_WorkbookSave(Wb_GenPart2)
End If
    *****

    'Map colums from PI sheet to MasterDataSheet
If flag Then
    Call z_ShNew("Log_8", "Begin")
    Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_6", "PMEC_VBA_MasterDataSet_7",
"Begin")
End If
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Map 33 attribute", "file: PMEC_VBA_PiDataSet_2 -
> PMEC_VBA_MasterDataSet_7", CStr(Now()), ""))
    'Define the attribute names to map into the Sh_to
    'Remark: SyngentaPortfolio=PI Syngenta Portfolio, Syngenta Portfolio=Activity Syngenta Portfolio
    Dim ColNames_to As Variant
    ' ColNames_to = Array( _
    '     "Portfolio Level 1", "Portfolio Level 2", "Portfolio Level 3", "SyngentaPortfolio", "PI Identifier", _
    '     "PI Title", "PI Status", "PI Label", "PI Comment", "PI Manager", _
    '     "PI Sponsor", "PI Syngenta Program", "PI Stage", "PI Scope", "PI Type", _
    '     "PI Sub Type", "PI Investment Category", "PI Customer", "PI Responsibility", "PI Geography", "PI
List of Regions", "PI List of Countries", _
    '     "PI list of Crops Group", "PI list of Crops", "PI Purchase Order", "PI Lead AI", "PI List of Active
Ingredients", _
    '     "PI Last Gate Passed", "PI Product Concept", "BC Total NPV", "BC Terminal Value NPV (10%
decline)", "BC First year of sales", "BC Sales Peak", "BC Required", "BC Business Case Author", "PI
Planned start", _
    '     "PI Planned finish", "PI Is confidential ?")
If flag Then
    'read attribute names
    Set Rng = z_RangeOfWb_AttributeNamesFromSheetToArray("GenerateReport", 4, Wb_from)
    ColNames_to = z_ReadAttributeNames_FromSheet_ToArray("GenerateReport", Rng, "ReadACol",
Wb_from)
End If
    'Write the attribute names to the first empty column in Sh_to
If flag Then
    Dim ColStart As Long

```

```

ColStart = z_ColSize(1, "PMEC_VBA_MasterDataSet_7") + 1
Call z_AddColNames(ColNames_to, "PMEC_VBA_MasterDataSet_7", 1, ColStart)
End If
If flag Then
'Define the attribute names in Sh_from
'Even though the attribute names in Sh_from and Sh_to may be called differently they must
'refer to the same attribute and they have to be in the same order in both arrays!!!!
'Dim ColNames_from As Variant
' ColNames_from = Array( _
'     "Portfolio Level 1", "Portfolio Level 2", "Portfolio Level 3", "Syngenta Portfolio", "PI Identifier", _
'     "PI Title", "PI Status", "PI Label", "PI Comment", "PI Manager", _
'     "PI Sponsor", "Syngenta Program", "PI Stage", "PI Scope", "PI Type", _
'     "PI Sub Type", "PI Investment Category", "PI Customer", "PI Responsibility", "PI Geography", "PI
List of Regions", "PI List of Countries", _
'     "PI list of Crops Group", "PI list of Crops", "PI Purchase Order", "PI Lead AI", "List of Active
Ingredients", _
'     "Last Gate Passed", "PI Product Concept", "Total NPV", "Terminal Value NPV (10% decline)",
"First year of sales", "Sales Peak", "BC Required", "PI Business Case Author", "Planned start", _
'     "Planned finish", "Is confidential ?")
Set Rng = z_RangeOfWb_AttributeNamesFromSheetToArray("GenerateReport", 3, Wb_from)
ColNames_from = z_ReadAttributeNames_FromSheet_ToArray("GenerateReport", Rng,
"ReadACol", Wb_from)

'check whether the column names exist in Sh_from
Dim sSh_from As String
sSh_from = "PMEC_VBA_PiDataSet_2"
'Call z_ColNamesExist(ColNames_from, sSh_from) '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!look into shmapcolumns()
'call the map function
'Call z_ShMapColumns(sSh_from, "PI Identifier", ColNames_from, "PMEC_VBA_MasterDataSet_7",
"PIIdentifier", ColNames_to, "Log_8")
Call z_ShMapColumns_FastVersion(sSh_from, "PI Identifier", ColNames_from,
"PMEC_VBA_MasterDataSet_7", _
    "PIIdentifier", ColNames_to, "Log_8")
Call z_WorkbookSave(Wb_GenPart2)
End If
'*****

'PI Grouping
'add new sheets
If flag Then
Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_7", "PMEC_VBA_MasterDataSet_8",
"Begin")
End If
If flag Then
Call z_Correct_ListOfPiGrouping("PMEC_VBA_MasterDataSet_8", "PIIdentifier", "PI List of PI
Grouping")
'MsgBox ("check pi grouping")
'Stop
'*****
'*****

'Crop Split for CPD and new also CPR
'add new sheets
End If
If flag Then

```



```

    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Map 9 cols for Crop Split", "file:
PMEC_VBA_MasterDataSet_9", CStr(Now()), ""))
    Call z_ShNew("Log_10_1", "Begin")
    Call z_ShNew("Log_10_2", "Begin")
    Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_8", "PMEC_VBA_MasterDataSet_9",
"Begin")
    Call z_WorkbookSave(Wb_GenPart2)
End If
    'add empty columns for the crop split and give column names (strategic crop, strategic_crop_flag
and those of the input sheet)
If flag Then
    Col_to = z_ColSize(1, "PMEC_VBA_MasterDataSet_9")
    'Dim ColNames_to As Variant
    ColNames_to = Array("PI Strategic Crop", "PI Strategic Crop flag", _
        "%Cereals", "%Corn", "%DFC", "%Rice", "%Soybean", "%Speciality", _
        "%Sugarcane", "%Vegetables", "%Lawn & Garden", "%Non-Crop")
End If
If flag Then
    Call z_AddColNames(ColNames_to, "PMEC_VBA_MasterDataSet_9", 1, Col_to + 1)
End If
*****
*****

'make sure you have a sheet with the new corp split CPD
Dim Wb_TemplateCropSplit As Workbook
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
WbName = "Template_CropSplit.xlsm"
Dim sSh_from_2 As String
sSh_from_2 = "CropSplit"
*****
*****

'open the CropSplit wb
If flag Then
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_TemplateCropSplit)
End If
    'move the Crop split sheet into the Wb_GenPart2
If flag Then
    Call z_CopySheetFromWb1ToWb2(sSh_from_2, Wb_TemplateCropSplit, Wb_GenPart2, "Begin")
    'Call z_CopySheetFromWb1ToWb2("OLD Crop Split CP", Wb_TemplateCropSplit, Wb_GenPart2,
"Begin")
End If
If flag Then
    'close Wb_CropSplitCPD, closes the active workbook and saves any changes
    Wb_TemplateCropSplit.Close False
End If
    'fill the crop split template
If flag Then
    Dim NameArrayCropPercent_template As Variant
    NameArrayCropPercent_template = Array("%Cereals", "%Corn", "%DFC", "%Rice", "%Soybean",
"%Speciality", _
        "%Sugarcane", "%Vegetables", "%Lawn & Garden", "%Non-Crop")

```

```

    Call z_fillCropSplitTemplate(sSh_from_2, "PI Identifier", NameArrayCropPercent_template,
"%Total", _
        "PMEC_VBA_MasterDataSet_9", "PIdentifier", "PI List of Strategic Crops", "PI Strategic Crop
Coefficient")
End If
If flag Then
    Call z_ExcelSessionWindowMinimized(Wb_GenPart2)
    Call z_ExcelSessionWindowNormal(Wb_GenPart2)
    MsgBox ("delete the lines without 100%")
    Stop
    Call z_RemoveLinesWithStrategicCropCoefficientNot100Percent("CropSplit", "PI Identifier",
"%Total", "Log_10_2")
    Stop
End If
'map the crop split % columns from sSh_from_2 into the MasterDataSet
If flag Then
    Call z_ShMapColumns_FastVersion(sSh_from_2, "PI Identifier", NameArrayCropPercent_template,
-
        "PMEC_VBA_MasterDataSet_9", "PIdentifier", NameArrayCropPercent_template,
"Log_10_1", , 1, 1)
End If
*****

'Perform the crop split
If flag Then
'Define the input parameters for the crop split function
Dim Pild_Col As Long
Dim CostCol_from_1 As Long
Dim CostCol_to_1 As Long
Dim CostCol_from_2 As Long
Dim CostCol_to_2 As Long
Dim StrategicCrop_Col As Long
Dim StrategicCropFlag_col As Long
Dim Crop_Col_from As Long
Dim Crop_Col_to As Long
Dim PL1_Col As Long
Dim PL2_Col As Long
Dim PL3_Col As Long
'CostCol_from_1 = z_GetColumnIndex("EAC SD Full Costs 2010", 1,
"PMEC_VBA_MasterDataSet_9")
CostCol_from_1 = z_GetColumnIndex("AC # Trials " & FirstYr, 1, "PMEC_VBA_MasterDataSet_9")
CostCol_to_1 = z_GetColumnIndex("EAC # Trials_c", 1, "PMEC_VBA_MasterDataSet_9")
CostCol_from_2 = z_GetColumnIndex("AC Full Costs " & FirstYr, 1, "PMEC_VBA_MasterDataSet_9")
CostCol_to_2 = z_GetColumnIndex("EAC Ext $ Direct Costs_c", 1, "PMEC_VBA_MasterDataSet_9")
StrategicCrop_Col = z_GetColumnIndex("PI Strategic Crop", 1, "PMEC_VBA_MasterDataSet_9")
StrategicCropFlag_col = z_GetColumnIndex("PI Strategic Crop flag", 1,
"PMEC_VBA_MasterDataSet_9")
Crop_Col_from = z_GetColumnIndex("%Cereals", 1, "PMEC_VBA_MasterDataSet_9")
Crop_Col_to = z_GetColumnIndex("%Non-Crop", 1, "PMEC_VBA_MasterDataSet_9")
Pild_Col = z_GetColumnIndex("PIdentifier", 1, "PMEC_VBA_MasterDataSet_9")
PL1_Col = z_GetColumnIndex("Portfolio Level 1", 1, "PMEC_VBA_MasterDataSet_9")
PL2_Col = z_GetColumnIndex("Portfolio Level 2", 1, "PMEC_VBA_MasterDataSet_9")
'check whether all columns have been found

```

```

    If CostCol_from_1 = 0 Or CostCol_to_1 = 0 Or CostCol_from_2 = 0 Or CostCol_to_2 = 0 Or
StrategicCrop_Col = 0 _
        Or StrategicCropFlag_col = 0 Or Crop_Col_from = 0 Or Crop_Col_to = 0 Or Pild_Col = 0 Or
PL1_Col = 0 _
        Or PL2_Col = 0 Then
        Stop
    End If
    'change the format
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Crop Split", "file: P MEC_VBA_MasterDataSet_9",
CStr(Now()), ""))
    Dim CropSplitRange As Range
    RowSize = z_RowSize(Pild_Col, "P MEC_VBA_MasterDataSet_9")
    Set CropSplitRange = Range(Cells(2, Crop_Col_from), Cells(RowSize, Crop_Col_to))
    CropSplitRange.Select
    Call z_TrimCells("P MEC_VBA_MasterDataSet_9", CropSplitRange)
    CropSplitRange.Select
    Selection.NumberFormat = "0.00"
    'Perform the crop split
    Application.ScreenUpdating = False
    Call z_CropSplit("P MEC_VBA_MasterDataSet_9", "Log_10_1", CostCol_from_1, CostCol_to_1,
CostCol_from_2, CostCol_to_2, _
        StrategicCrop_Col, StrategicCropFlag_col, Crop_Col_from, Crop_Col_to, Pild_Col, PL1_Col,
PL2_Col)
    Application.ScreenUpdating = True
    Call z_WorkbookSave(Wb_GenPart2)
End If
    'make changes to the column strategic crop (those with flag=3)
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Strategic Crop for Seeds and BusinessDev", "file:
P MEC_VBA_MasterDataSet_9", CStr(Now()), ""))
    Pild_Col = z_GetColumnIndex("Pildentifier", 1, "P MEC_VBA_MasterDataSet_9")
    StrategicCrop_Col = z_GetColumnIndex("PI Strategic Crop", 1, "P MEC_VBA_MasterDataSet_9")
    'Call z_InsertColumn("P MEC_VBA_MasterDataSet_9", StrategicCrop_Col,
"P MEC_VBA_MasterDataSet_9", StrategicCrop_Col)
    'Cells(1, StrategicCrop_Col + 1) = "Strategic Crop 2"
    PL1_Col = z_GetColumnIndex("Portfolio Level 1", 1, "P MEC_VBA_MasterDataSet_9")
    PL2_Col = z_GetColumnIndex("Portfolio Level 2", 1, "P MEC_VBA_MasterDataSet_9")
    PL3_Col = z_GetColumnIndex("Portfolio Level 3", 1, "P MEC_VBA_MasterDataSet_9")
    StrategicCropCoeff_Col = z_GetColumnIndex("PI Strategic Crop Coefficient", 1,
"P MEC_VBA_MasterDataSet_9")
    Call z_StrategicCrop_ChangeEntriesForSEEDS("P MEC_VBA_MasterDataSet_9", Pild_Col,
StrategicCrop_Col, PL1_Col, PL2_Col, PL3_Col)
    Call z_StrategicCrop_ChangeEntriesForBUSINESSDEV("P MEC_VBA_MasterDataSet_9", Pild_Col,
StrategicCrop_Col, PL1_Col, PL2_Col, PL3_Col)
    'save the wb
    Call z_WorkbookSave(Wb_GenPart2)
End If
If flag Then
    '=====
    '=====
    '-open new process

```

```

'-open this workbook in the new process (read only)
'-set on time to run this workbook in the new process
'-close this workbook (read/write)
'-close this process
'-wait until on time starts the procedure resume
'-resume sets this workbook in the new process to read/write
'-resume calls the sub m_ReStart_Main_GenerateReport
'-the flag_rerun_Generation_Part2To4 set to 1 resumes the code here
Call m_Main_MemoryLeakWorkaround_2
'=====
'=====
End If
*****
*****

'Adds a new workbook RD_MasterDataSet because of memory problems
*****
*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
Dim Wb_GenPart3 As Workbook
If flag = 0 Then
    If flag_rerun_Generation_Part3To4 = 1 Then
        flag = 1
    End If
End If
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part3_" & wbdate &
reportversion & ".xlsb"
Else
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part3_" & wbdate &
reportversion & ".xlsb"
    If flag_rerun_Generation_Part4To4 = 1 Then 'If flag_reopen_wb Then
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart3)
    End If
End If
If flag Then
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, Wb_GenPart3)
End If
If flag Then
    'copy the last datasheet into the new workbook
    'Call z_MoveSheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart2,
Wb_GenPart3, "Begin")
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_9", Wb_GenPart2, Wb_GenPart3,
"Begin")
    'in case they have already been deleted in a previous run
On Error Resume Next
    Call z_ShDelete("Sheet1", Wb_GenPart3)
    Call z_ShDelete("Sheet2", Wb_GenPart3)
    Call z_ShDelete("Sheet3", Wb_GenPart3)
On Error GoTo 0

```

```

    Call z_WorkbookSave(Wb_GenPart3)
End If
If flag Then
    'close Wb_GenPart2, closes the active workbook and saves any changes
    Wb_GenPart2.Close True
End If
    *****

    'add the concatenated string Pild&PiTitle and ActivityId&TaskTitle
    'add new sheets
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Make concatenated string Pild+Title, Tkld+Title",
"file: P MEC_VBA_MasterDataSet_10", CStr(Now()), ""))
    Call z_ShNew("Log_11", "Begin")
    Call z_ShNewFlatValueCopy("P MEC_VBA_MasterDataSet_9", "P MEC_VBA_MasterDataSet_10",
"Begin")
End If
If flag Then
    'add two columns
    Col_to = z_ColSize(1, "P MEC_VBA_MasterDataSet_10") + 1
    Call z_InsertEmptyCols("P MEC_VBA_MasterDataSet_10", 2, Col_to) 'not necessary
    Call z_AddColNames(Array("Pildentifier&PiTitle", "ActivityIdentifier&TaskTitle"),
"P MEC_VBA_MasterDataSet_10", 1, Col_to)
End If
    'Pild&PiTitle
If flag Then
    Dim PiTitle_Col As Long
    Pild_Col = z_GetColumnIndex("Pildentifier", 1, "P MEC_VBA_MasterDataSet_10")
    PiTitle_Col = z_GetColumnIndex("PI Title", 1, "P MEC_VBA_MasterDataSet_10")
    'ActivityId&TaskTitle
    Dim TaskId_Col As Long
    Dim TaskTitle_Col As Long
    TaskId_Col = z_GetColumnIndex("ActivityIdentifier", 1, "P MEC_VBA_MasterDataSet_10")
    TaskTitle_Col = z_GetColumnIndex("Task Title", 1, "P MEC_VBA_MasterDataSet_10") 'check the
name!!!
End If
    'fill the columns
If flag Then
    RowSize = z_RowSize(Pild_Col, "P MEC_VBA_MasterDataSet_10")
    Cells(1, Col_to).Select
    For Row = 2 To RowSize
        Cells(Row, Col_to) = Cells(Row, Pild_Col) & " : " & Cells(Row, PiTitle_Col)
        Cells(Row, Col_to + 1) = Cells(Row, TaskId_Col) & " : " & Cells(Row, TaskTitle_Col)
    Next Row
End If
    *****

    'add the RBS
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Add RBS into 5 cols", "file:
P MEC_VBA_MasterDataSet_11", CStr(Now()), ""))
    Call z_ShNew("Log_12", "Begin")

```

```

    Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_10", "PMEC_VBA_MasterDataSet_11",
"Begin")
End If
*****

*****

'make sure you have an RBS extract available
Dim Wb_RBS As Workbook
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
WbName = "RBSTable.xlsb"
Dim sSh_from_3 As String
sSh_from_3 = "RBS Table"
*****

*****

'open the RBS extract
If flag Then
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_RBS)
End If
'move the RBS extract into the Wb_GenPart2
If flag Then
    Call z_CopySheetFromWb1ToWb2(sSh_from_3, Wb_RBS, Wb_GenPart3, "Begin")
End If
If flag Then
    'close Wb_RBS, closes the active workbook and saves any changes
    Wb_RBS.Close True
End If
'add five columns
If flag Then
    Dim Resource_Col As Long
    Dim Unit_Col As Long
    Resource_Col = z_GetColumnIndex("Resource", 1, "PMEC_VBA_MasterDataSet_11")
    Unit_Col = z_GetColumnIndex("Unit", 1, "PMEC_VBA_MasterDataSet_11")
    Col_to = z_ColSize(1, "PMEC_VBA_MasterDataSet_11") + 1
End If
If flag Then
    Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_11", 5, Col_to)
    Call z_AddColNames(Array("RBS Level 1", "RBS Level 2", "RBS Level 3", "Cost Unit", "Resource
Name"), _
        "PMEC_VBA_MasterDataSet_11", 1, Col_to)
End If
If flag Then
    Dim RBS1_Col As Long
    Dim RBS2_Col As Long
    Dim RBS3_Col As Long
    Dim CostUnit_col As Long
    Dim ResourceName_col As Long
    RBS1_Col = z_GetColumnIndex("RBS Level 1", 1, "PMEC_VBA_MasterDataSet_11")
    RBS2_Col = z_GetColumnIndex("RBS Level 2", 1, "PMEC_VBA_MasterDataSet_11")
    RBS3_Col = z_GetColumnIndex("RBS Level 3", 1, "PMEC_VBA_MasterDataSet_11")
    CostUnit_col = z_GetColumnIndex("Cost Unit", 1, "PMEC_VBA_MasterDataSet_11")
    ResourceName_col = z_GetColumnIndex("Resource Name", 1, "PMEC_VBA_MasterDataSet_11")
    Resource_Col = z_GetColumnIndex("Resource", 1, "PMEC_VBA_MasterDataSet_11")
End If

```

```

'add the Cost Unit and Resource Name
If flag Then
    RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_11")
    Range(Cells(2, Unit_Col), Cells(RowSize, Unit_Col)).Select
    'MsgBox ("correct the function")
    'Stop
    Dim Unit_Resource As Variant
    For Each Unit In Selection.Cells
        Unit_Resource = Split(Unit, "___", -1)
        If UBound(Unit_Resource) > 0 Then
            Range(Unit.Offset(0, CostUnit_col - Unit_Col), Unit.Offset(0, ResourceName_col - Unit_Col)) _
                = Unit_Resource
        Else
            rw = Unit.Row
            Unit.Offset(0, CostUnit_col - Unit_Col) = Unit_Resource(0)
            Unit.Offset(0, ResourceName_col - Unit_Col) = Unit.Offset(0, Resource_Col - Unit_Col)
        End If
    Next
End If
'map the RBS level 1 to 3, Key_to=ResourceName_col
If flag Then
    Call z_AddRBSLevels(sSh_from_3, "PMEC_VBA_MasterDataSet_11", Pild_Col, ResourceName_col,
RBS1_Col, RBS2_Col, RBS3_Col)
End If
'map the RBS level 1 to 3, Key_to=Resource_col
If flag Then
    'MsgBox ("check whether still used")
    'Stop
    Call z_AddRBSLevels(sSh_from_3, "PMEC_VBA_MasterDataSet_11", Pild_Col, Resource_Col,
RBS1_Col, RBS2_Col, RBS3_Col)
End If
'*****

'for projects with confidential=Yes replace the Text from the following attributes
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Overwrite 11 cols of Confidential Pis", "file:
PMEC_VBA_MasterDataSet_12", CStr(Now()), ""))
    Call z_ShNew("Log_13", "Begin")
    Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_11", "PMEC_VBA_MasterDataSet_12",
"Begin")
End If
If flag Then
    'define the attributes whose values must be deleted for confidential projects
    Dim ColNameConfidential_Arr As Variant
    ' ColNameConfidential_Arr = Array("PI Title", "PIdentifier&PiTitle", "PI Label", "PI Comment", "PI
Scope", _
    ' "PI Lead AI", "Description", "Activity Comment", "PI List of Active Ingredients", "Task Title",
"ActivityIdentifier&TaskTitle")
    '
    Set Rng = z_RangeOfWb_AttributeNamesFromSheetToArray("GenerateReport", 5, Wb_from)
    ColNameConfidential_Arr = z_ReadAttributeNames_FromSheet_ToArray("GenerateReport", Rng,
"ReadACol", Wb_from)

```

```

'fill the array PildConfidential_Arr with all confidential Pilds
'Dim PildConfidential_Arr As Variant
ReDim PildConfidential_Arr(0 To 1000) As Variant
Dim IsConfidential_Col As Long
IsConfidential_Col = z_GetColumnIndex("PI Is confidential ?", 1, "PMEC_VBA_MasterDataSet_12")
Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_12")
RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_12")
iter = 0
For Row = 2 To RowSize
    If Cells(Row, IsConfidential_Col) = True Or Cells(Row, IsConfidential_Col) = "YES" Then
        If iter = 0 Then
            PildConfidential_Arr(iter) = Cells(Row, Pild_Col)
            iter = iter + 1
        ElseIf PildConfidential_Arr(iter - 1) <> Cells(Row, 1) Then
            PildConfidential_Arr(iter) = Cells(Row, Pild_Col)
            iter = iter + 1
        End If
    End If
Next Row
'Redim the array, remove the empty indexes
For iter = 1 To UBound(PildConfidential_Arr)
    If PildConfidential_Arr(iter) = Empty Then
        ReDim Preserve PildConfidential_Arr(0 To iter - 1)
        Exit For
    End If
Next iter
'delete in all confidential projects the confidential information
For Pilter = LBound(PildConfidential_Arr) To UBound(PildConfidential_Arr)
    Dim Pi_i As String
    Pi_i = PildConfidential_Arr(Pilter)
    'select all the rows for which the value in the column Pild_Col equals Pi_i
    Call z_SelectMultiple("PMEC_VBA_MasterDataSet_12", Pild_Col, Pi_i)
    'replace the value in the columns
    For Collter = LBound(ColNameConfidential_Arr) To UBound(ColNameConfidential_Arr)
        Dim ColName_i As String
        Dim Col_i As Long
        ColName_i = ColNameConfidential_Arr(Collter)
        Col_i = z_GetColumnIndex(ColName_i, 1, "PMEC_VBA_MasterDataSet_12")
        If ColName_i = "Pildentifier&PiTitle" Then
            'change the value in the cells in Collter
            Selection.Columns(Col_i) = Pi_i & " - " & "Confidential" & " : " & Pi_i 'Empty
        Else
            'change the value in the cells in Collter
            Selection.Columns(Col_i) = "Confidential" & " : " & Pi_i 'Empty
        End If
    Next Collter
Next Pilter
End If
'Map the building blocks and strategic pillars
*****

If flag Then
    Call z_ExcelSessionWindowMinimized(Wb_GenPart3)
    Call z_ExcelSessionWindowNormal(Wb_GenPart3)

```



```

MsgBox ("move the sheet_From into this Wb check the names and map building blocks and
strategic pillars")
Stop
*****

*****

'make sure you have the wb available
Dim Wb_SPandBB As Workbook
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
WbName = "SPandBB.xlsm"
Dim sSh_from_4 As String
sSh_from_4 = "SPandBB"
*****

*****

End If
If flag Then
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_SPandBB)
End If
'move
If flag Then
    Call z_CopySheetFromWb1ToWb2(sSh_from_4, Wb_SPandBB, Wb_GenPart3, "Begin")
End If
If flag Then
    'close
    Wb_SPandBB.Close True
End If
If flag Then
    'make
    Col_to = z_ColSize(1, "PMEC_VBA_MasterDataSet_12") + 1
    Call z_AddColNames(Array("Pildentifier & PI Strategic Crop", "PI Building Blocks", "PI Strategic
Pillar"), "PMEC_VBA_MasterDataSet_12", 1, Col_to)
    'fill the colum
    Dim Col_PildStrCrop As Long
    Dim Col_StrCrop As Long
    Col_PildStrCrop = z_GetColumnIndex("Pildentifier & PI Strategic Crop", 1,
"PMEC_VBA_MasterDataSet_12")
    Col_StrCrop = z_GetColumnIndex("PI Strategic Crop", 1, "PMEC_VBA_MasterDataSet_12")
    Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_12")
    RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_12")
    Cells(1, Col_PildStrCrop).Select
    For Row = 2 To RowSize
        Cells(Row, Col_PildStrCrop) = Cells(Row, Pild_Col) & "@" & Cells(Row, Col_StrCrop)
    Next Row
End If
If flag Then
    'map
    ColNames_from = Array("PI Building Blocks", "PI Strategic Pillar")
    ColNames_to = ColNames_from
    sSh_from = "SPandBB"
    Call z_ShMapColumns_FastVersion(sSh_from, "Pildentifier & PI Strategic Crop", ColNames_from,
"PMEC_VBA_MasterDataSet_12", "Pildentifier & PI Strategic Crop", ColNames_to)
    Call z_WorkbookSave(Wb_GenPart3)
End If

```

```

'map Syngenta Program = ICS (Yes/No)
Dim Wb_SynPrograms As Workbook
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
WbName = "SyngentaProgrammes.xlsx"
Dim sSh_from_5 As String
sSh_from_5 = "ICS_Programmes"
*****
*****

If flag Then
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_SynPrograms)
End If
'move
If flag Then
    Call z_CopySheetFromWb1ToWb2(sSh_from_5, Wb_SynPrograms, Wb_GenPart3, "Begin")
End If
If flag Then
    'close
    Wb_SynPrograms.Close True
End If
If flag Then
    MsgBox ("ICS? : Methodology with Titel and Program is turned off. Map ICS_Pls.xlsb at the end of
this procedure")
    Stop
    Col_to = z_ColSize(1, "PMEC_VBA_MasterDataSet_12") + 1
    Call z_AddColNames(Array("ICS?"), "PMEC_VBA_MasterDataSet_12", 1, Col_to)
    Col_IsICS_to = z_GetColumnIndex("ICS?", 1, "PMEC_VBA_MasterDataSet_12")
    Col_SynProg_to = z_GetColumnIndex("PI Syngenta Program", 1, "PMEC_VBA_MasterDataSet_12")
    Dim Pild_Col_to As Long
    Pild_Col_to = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_12")
    RowSize_to = z_RowSize(Pild_Col_to, "PMEC_VBA_MasterDataSet_12")

    Col_SynProg_from = z_GetColumnIndex("Syngenta Program", 1, sSh_from_5)
    Col_SynProgTitle_from = z_GetColumnIndex("Program Title", 1, sSh_from_5)
    Dim ProgramId_Col_from As Long
    ProgramId_Col_from = z_GetColumnIndex("Program Identifier", 1, sSh_from_5)
    RowSize_from = z_RowSize(ProgramId_Col_from, sSh_from_5)

    Dim Col_PiTitle As Long
    Col_PiTitle = z_GetColumnIndex("PI Title", 1, "PMEC_VBA_MasterDataSet_12")

    Dim Row2_to As Long
    ' For Row2_to = 2 To RowSize_to
    '     flag_ICS = False
    '     'check extract
    '     If Sheets("PMEC_VBA_MasterDataSet_12").Cells(Row2_to, Col_SynProg_to) <> Empty Then
    '         For Row_from = 2 To RowSize_from
    '             If Sheets("PMEC_VBA_MasterDataSet_12").Cells(Row2_to, Col_SynProg_to) =
Sheets(sSh_from_5).Cells(Row_from, Col_SynProgTitle_from) Then
    '                 Sheets("PMEC_VBA_MasterDataSet_12").Cells(Row2_to, Col_IsICS_to) = "YES"
    '                 flag_ICS = True
    '                 Exit For
    '             End If

```

```

'    Next
'    End If
'    'check pi title
'    If Sheets("PMEC_VBA_MasterDataSet_12").Cells(Row2_to, Col_PiTitle) Like "ICS*" Then
'        Sheets("PMEC_VBA_MasterDataSet_12").Cells(Row2_to, Col_PiTitle).Activate
'        Sheets("PMEC_VBA_MasterDataSet_12").Cells(Row2_to, Col_IsICS_to).Activate
'        Sheets("PMEC_VBA_MasterDataSet_12").Cells(Row2_to, Col_IsICS_to) = "YES"
'        flag_ICS = True
'    End If
'    'if not yes set to no
'    If flag_ICS = False Then
'        Sheets("PMEC_VBA_MasterDataSet_12").Cells(Row2_to, Col_IsICS_to) = "NO"
'    End If
'    Next
'    Call z_WorkbookSave(Wb_GenPart3)
End If

```

```

If flag Then
    MsgBox ("Some coding to do here and add in the Attribute Sheet at position Set Rng =
z_RangeOfWb_AttributeNamesFromSheetToArray(GenerateReport, 6, Wb_from)")
    ""ICS?" <- From Jaspers spreadsheet
    ""BIO-CONTROLS?" <- PI List of PI Grouping
    ""CE?" <- PI List of PI Grouping
    ""PER?" <- PI List of PI Grouping
    ""Adjacent Technology?" <- From Franz Lanter's spreadsheet
End If

```

```

If flag Then
'=====
'=====
'open new process
'open this workbook in the new process (read only)
'set on time to run this workbook in the new process
'close this workbook (read/write)
'close this process
'wait until on time starts the procedure resume
'resume sets this workbook in the new process to read/write
'resume calls the sub m_ReStart_Main_GenerateReport
'the flag_rerun_Generation_Part2To4 set to 1 resumes the code here
Call m_Main_MemoryLeakWorkaround_3
'=====
'=====

```

```

End If
'*****
'*****
'Adds a new workbook RD_MasterDataSet because of memory problems
'*****
'*****
'Add a new workbook and move the MasterDataSet and the PiDataSet into it
Dim Wb_GenPart4 As Workbook
If flag = 0 Then
    If flag_rerun_Generation_Part4To4 = 1 Then
        flag = 1
    End If

```

```

    End If
End If
If flag Then
Else
    If flag_reopen_wb Then
        Wb_GenPart3.Close False
    End If
End If
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part4_" & wbdate &
reportversion & ".xlsb"
Else
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion &
folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part4_" & wbdate &
reportversion & ".xlsb"
    If flag_reopen_wb Then
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart4)
    End If
End If
If flag Then
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, Wb_GenPart4)
End If
If flag Then
    'copy the last datasheet into the new workbook
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_12", Wb_GenPart3,
Wb_GenPart4, "Begin")
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", Wb_GenPart4)
        Call z_ShDelete("Sheet2", Wb_GenPart4)
        Call z_ShDelete("Sheet3", Wb_GenPart4)
    On Error GoTo 0
    Call z_WorkbookSave(Wb_GenPart4)
End If
If flag Then
    'close Wb_GenPart2, closes the active workbook and saves any changes
    Wb_GenPart3.Close True
End If

*****

'copy only the needed columns into a new sheet
'the array determines the order of the columns
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Copy 130 cols", "file:
PMEC_VBA_MasterDataSet_13", CStr(Now()), ""))
    Call z_ShNew("Log_14", "Begin")
    Call z_ShNew("PMEC_VBA_MasterDataSet_13", "Begin")
    'create the ColName_Arr with the function z_CreateArrayInput_AttributeNames
    'Call z_CreateArrayInput_AttributeNames

```

```

'Remark: SyngentaPortfolio=PI Syngenta Portfolio, Syngenta Portfolio=Activity Syngenta Portfolio
Dim ColName_Arr As Variant

Set Rng = z_RangeOfWb_AttributeNamesFromSheetToArray("GenerateReport", 6, Wb_from)
ColName_Arr = z_ReadAttributeNames_FromSheet_ToArray("GenerateReport", Rng, "ReadACol",
Wb_from)

For iter = LBound(ColName_Arr) To UBound(ColName_Arr)
    Dim Col_to_i As Long
    Dim Col_from_i As Long
    'Dim ColName_i As String
    Col_to_i = iter + 1
    ColName_i = ColName_Arr(iter)
    Col_from_i = z_GetColumnIndex(ColName_i, 1, "PMEC_VBA_MasterDataSet_12")
'Pildentifier&PiTitle
    Call z_CopyColumn("PMEC_VBA_MasterDataSet_12", Col_from_i,
"PMEC_VBA_MasterDataSet_13", Col_to_i)
    Next iter
    Call z_WorkbookSave(Wb_GenPart4)
End If
*****

'Date corrections
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: Date correction, renaming ...", "file:
PMEC_VBA_MasterDataSet_13", CStr(Now()), ""))
    'change the date format
    Dim date_Arr As Variant
    'date_arr = Array("Duration", "Planned start", "Expected finish export", "Actual start", "Actual
Finish export", "PI Planned start", "PI Planned finish")
    Set Rng = z_RangeOfWb_AttributeNamesFromSheetToArray("GenerateReport", 7, Wb_from)
    date_Arr = z_ReadAttributeNames_FromSheet_ToArray("GenerateReport", Rng, "ReadACol",
Wb_from)

    Call z_ChgDateFormat("PMEC_VBA_MasterDataSet_13", date_Arr, 0)
    'make date object where not already
    'Dim Rng As Range
    Dim Date_ColNames As Variant
    'Dim Date_Col_i As Long
    Dim Date_ColName_i As String
    Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_13")
    RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_13")
    'Date_ColNames = Array("PI Planned start", "PI Planned finish", "Planned start", "Expected finish
export", "Actual start", "Actual Finish export")
    Set Rng = z_RangeOfWb_AttributeNamesFromSheetToArray("GenerateReport", 8, Wb_from)
    Date_ColNames = z_ReadAttributeNames_FromSheet_ToArray("GenerateReport", Rng,
"ReadACol", Wb_from)

    For iter = LBound(Date_ColNames) To UBound(Date_ColNames)
        Date_ColName_i = Date_ColNames(iter)
        Date_Col_i = z_GetColumnIndex(Date_ColName_i, 1, "PMEC_VBA_MasterDataSet_13")
        Set Rng = Range(Cells(1, Date_Col_i), Cells(RowSize, Date_Col_i))
        Call z_DateCorrection(Rng)
    
```

```

Next iter
'Date_Arr = Array("ActualStart", "ActualFinishExport", "StartNoEarlierThan",
"FinishNoLaterThanExport")
'Call z_ChgDateFormat("PMEC_VBA_MasterDataSet_9", Date_Arr, 1)
Call z_WorkbookSave(Wb_GenPart4)
End If
'for value attributes:replace all blanks with "0"
'for non value attributes: replace all blanks with "(blank)"
*****

If flag Then
Call z_replaceEmptyCellsWithZero("PMEC_VBA_MasterDataSet_13", "Like", "EAC*")
Call z_replaceEmptyCellsWithBLANK("PMEC_VBA_MasterDataSet_13", "NotLike", "EAC*")
Call z_WorkbookSave(Wb_GenPart4)
End If

*****

'rename some attributes (remove export)
'Remark: SyngentaPortfolio=PI Syngenta Portfolio, Syngenta Portfolio=Activity Syngenta Portfolio
If flag Then
'Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Terminal Value NPV (10% decline)",
"BC Terminal Value (10% decline)")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "MyResourceId", "RsIdentifier")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "SyngentaPortfolio", "PI Syngenta
Portfolio")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Expected finish export", "Activity
Expected Finish")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Actual Finish export", "Activity
Actual Finish")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Task Title", "TK Task Title")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Description", "Activity Description")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Task Contact", "TK Task Contact")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Task Location", "TK Task Location")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Task Status", "TK Task Status")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Task Customer", "TK Task
Customer")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Duration", "Activity Duration")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Activity type", "Activity Type")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Planned Start", "Activity Planned
Start")
Call z_RenameAttribute("PMEC_VBA_MasterDataSet_13", 1, "Actual Start", "Activity Actual Start")
End If
*****

If flag Then
Call z_WorkbookSave(Wb_GenPart4)
End If
If flag Then
'close Wb_GenPart3, closes the active workbook and saves any changes
Wb_GenPart4.Close True
Else
If flag_reopen_wb Then
Wb_GenPart4.Close False
End If
End If

```

```

Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
    Array("Main_CreateReport", "End", "task: ", CStr(Now()), ""))
End Sub

```

```

Sub m_GenerateReport_PlannedSales()

```

```

    Dim Wb As Workbook
    Dim Sh_from As Worksheet
    Dim Row_from As Long
    Dim col_Pild_from As Long
    Dim Col_Unit_from As Long
    Dim Col_StartDate_from As Long
    Dim Col_Quantity_from As Long
    Dim Sh_to As Worksheet
    Dim Row_to As Long
    Dim Col_Pild_to As Long
    Dim Col_PlannedSales_to As Long
    Dim Pild_val As String
    Dim Pild_next_val As String
    Dim Unit_from_val As String
    Dim Year_from_val As String
    Dim Quantity_from_val As String
    Dim PlannedSales_to As String

```

```

    MsgBox ("open the download workbook")

```

```

    Stop

```

```

    Set Wb = Workbooks("CONFIDENTIAL_SmC_Download_2012_09_20_V1-0.xlsm")

```

```

    Wb.Activate

```

```

    Call z_ShNew("SmC_PlannedSales", "Begin")

```

```

    Set Sh_to = Wb.Sheets("SmC_PlannedSales")

```

```

    Sh_to.Activate

```

```

    Dim FirstYr As Integer

```

```

    FirstYr = year(Now()) - 7

```

```

    Dim Col_Start_to As Long

```

```

    Col_Start_to = 1

```

```

    Call z_AddColNames(Array("Pildentifier", "Planned Sales " & FirstYr & " SALES_STANDALONE",
"Planned Sales " & FirstYr + 1 & " SALES_STANDALONE", "Planned Sales " & FirstYr + 2 & "
SALES_STANDALONE", "Planned Sales " & FirstYr + 3 & " SALES_STANDALONE", "Planned Sales " &
FirstYr + 4 & " SALES_STANDALONE", _
    "Planned Sales " & FirstYr + 5 & " SALES_STANDALONE", "Planned Sales " & FirstYr + 6 & "
SALES_STANDALONE", "Planned Sales " & FirstYr + 7 & " SALES_STANDALONE", "Planned Sales " &
FirstYr + 8 & " SALES_STANDALONE", "Planned Sales " & FirstYr + 9 & " SALES_STANDALONE",
"Planned Sales " & FirstYr + 10 & " SALES_STANDALONE", _
    "Planned Sales " & FirstYr + 11 & " SALES_STANDALONE", "Planned Sales " & FirstYr + 12 & "
SALES_STANDALONE", "Planned Sales " & FirstYr + 13 & " SALES_STANDALONE", "Planned Sales " &
FirstYr + 14 & " SALES_STANDALONE", "Planned Sales " & FirstYr + 15 & " SALES_STANDALONE",
"Planned Sales " & FirstYr + 16 & " SALES_STANDALONE", _
    "Planned Sales " & FirstYr + 17 & " SALES_STANDALONE", "Planned Sales " & FirstYr + 18 & "
SALES_STANDALONE", "Planned Sales " & FirstYr + 19 & " SALES_STANDALONE", "Planned Sales " &
FirstYr + 20 & " SALES_STANDALONE", "Planned Sales " & FirstYr + 21 & " SALES_STANDALONE",
"Planned Sales " & FirstYr + 22 & " SALES_STANDALONE", _

```

```

        "Planned Sales " & FirstYr + 23 & " SALES_STANDALONE", "Planned Sales " & FirstYr + 24 & "
SALES_STANDALONE", "Planned Sales " & FirstYr + 25 & " SALES_STANDALONE", "Planned Sales " &
FirstYr + 26 & " SALES_STANDALONE", "Planned Sales " & FirstYr + 27 & " SALES_STANDALONE",
"Planned Sales " & FirstYr + 28 & " SALES_STANDALONE"), _
        Sh_to.name, 1, Col_Start_to, , "Yes")
        Col_Start_to = z_ColSize(1, Sh_to.name) + 1
        Call z_AddColNames(Array("Planned Sales " & FirstYr & " SALES_CANNIBALIZATION", "Planned
Sales " & FirstYr + 1 & " SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 2 & "
SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 3 & " SALES_CANNIBALIZATION", "Planned
Sales " & FirstYr + 4 & " SALES_CANNIBALIZATION", _
        "Planned Sales " & FirstYr + 5 & " SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 6 & "
SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 7 & " SALES_CANNIBALIZATION", "Planned
Sales " & FirstYr + 8 & " SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 9 & "
SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 10 & " SALES_CANNIBALIZATION", _
        "Planned Sales " & FirstYr + 11 & " SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 12 & "
SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 13 & " SALES_CANNIBALIZATION", "Planned
Sales " & FirstYr + 14 & " SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 15 & "
SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 16 & " SALES_CANNIBALIZATION", _
        "Planned Sales " & FirstYr + 17 & " SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 18 & "
SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 19 & " SALES_CANNIBALIZATION", "Planned
Sales " & FirstYr + 20 & " SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 21 & "
SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 22 & " SALES_CANNIBALIZATION", _
        "Planned Sales " & FirstYr + 23 & " SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 24 & "
SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 25 & " SALES_CANNIBALIZATION", "Planned
Sales " & FirstYr + 26 & " SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 27 & "
SALES_CANNIBALIZATION", "Planned Sales " & FirstYr + 28 & " SALES_CANNIBALIZATION"), _
        Sh_to.name, 1, Col_Start_to, , "Yes")
        Col_Start_to = z_ColSize(1, Sh_to.name) + 1
        Call z_AddColNames(Array("Planned Sales " & FirstYr & " SALES_INCREMENTAL", "Planned Sales "
& FirstYr + 1 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 2 & " SALES_INCREMENTAL",
"Planned Sales " & FirstYr + 3 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 4 & "
SALES_INCREMENTAL", _
        "Planned Sales " & FirstYr + 5 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 6 & "
SALES_INCREMENTAL", "Planned Sales " & FirstYr + 7 & " SALES_INCREMENTAL", "Planned Sales " &
FirstYr + 8 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 9 & " SALES_INCREMENTAL",
"Planned Sales " & FirstYr + 10 & " SALES_INCREMENTAL", _
        "Planned Sales " & FirstYr + 11 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 12 & "
SALES_INCREMENTAL", "Planned Sales " & FirstYr + 13 & " SALES_INCREMENTAL", "Planned Sales " &
FirstYr + 14 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 15 & " SALES_INCREMENTAL",
"Planned Sales " & FirstYr + 16 & " SALES_INCREMENTAL", _
        "Planned Sales " & FirstYr + 17 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 18 & "
SALES_INCREMENTAL", "Planned Sales " & FirstYr + 19 & " SALES_INCREMENTAL", "Planned Sales " &
FirstYr + 20 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 21 & " SALES_INCREMENTAL",
"Planned Sales " & FirstYr + 22 & " SALES_INCREMENTAL", _
        "Planned Sales " & FirstYr + 23 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 24 & "
SALES_INCREMENTAL", "Planned Sales " & FirstYr + 25 & " SALES_INCREMENTAL", "Planned Sales " &
FirstYr + 26 & " SALES_INCREMENTAL", "Planned Sales " & FirstYr + 27 & " SALES_INCREMENTAL",
"Planned Sales " & FirstYr + 27 & " SALES_INCREMENTAL"), _
        Sh_to.name, 1, Col_Start_to, , "Yes")
        Sh_to.Rows(1).RowHeight = 40
        Sh_to.Rows(1).WrapText = True
        Sh_to.Rows(1).Interior.Color = RGB(220, 220, 220)
        Sh_to.Cells(2, 1).Activate

```



```

Row_to = 2
'iterate through all sheets
For Each Sh_from In Wb.Worksheets
    If left(Sh_from.name, 5) = "SmC_A" Then
        'Sh_from.Activate
        col_Pild_from = z_GetColumnIndex("Task or WBS element", 1, Sh_from.name)
        Col_Unit_from = z_GetColumnIndex("Unit", 1, Sh_from.name)
        Col_StartDate = z_GetColumnIndex("Start date", 1, Sh_from.name)
        Col_Quantity_from = z_GetColumnIndex("Quantity", 1, Sh_from.name)
        RowSize = z_RowSize(col_Pild_from, Sh_from.name)
        'iterate through all rows
        For Row_from = 2 To RowSize
            'Sh_from.Activate
            Pild_val = Sh_from.Cells(Row_from, col_Pild_from)
            Pild_next_val = Sh_from.Cells(Row_from + 1, col_Pild_from)
            Unit_from_val = Sh_from.Cells(Row_from, Col_Unit_from)
            Year_from_val = Mid(Sh_from.Cells(Row_from, Col_StartDate), 7, 4)
            Quantity_from_val = Sh_from.Cells(Row_from, Col_Quantity_from)
            PlannedSales_to = "Planned Sales " & Year_from_val & " " & Unit_from_val
            If Quantity_from_val <> Empty Then
                'Sh_to.Activate
                Col_PlannedSales_to = z_GetColumnIndex(PlannedSales_to, 1, Sh_to.name)
                If Col_PlannedSales_to = Empty Then
                    Stop
                End If
                If Pild_val Like "PI*" And Unit_from_val <> Empty Then
                    If Pild_val = Pild_next_val Then
                        'write
                        'Sh_to.Cells(Row_to, Col_PlannedSales_to).Activate
                        Sh_to.Cells(Row_to, Col_PlannedSales_to).Value = Quantity_from_val
                        Sh_to.Cells(Row_to, Col_PlannedSales_to).Interior.Color = RGB(216, 228, 188)
                    Else
                        'write and iterate
                        'Sh_to.Cells(Row_to, Col_PlannedSales_to).Activate
                        Sh_to.Cells(Row_to, Col_PlannedSales_to).Value = Quantity_from_val
                        Sh_to.Cells(Row_to, Col_PlannedSales_to).Interior.Color = RGB(216, 228, 188)
                        Col_Pild_to = z_GetColumnIndex("Pildentifier", 1, Sh_to.name)
                        Sh_to.Cells(Row_to, Col_Pild_to).Activate
                        Sh_to.Cells(Row_to, Col_Pild_to).Value = Pild_val
                        Row_to = Row_to + 1
                    End If
                End If
            End If
        Next Row_from
    End If
Next Sh_from
Cells(1, 89) = "Incremental Peak Sales"
MsgBox ("sum formel in incremental spalten manuell einfügen in BH2: =B2-AE2")
MsgBox ("Für peak sales max formel in incremental spalten manuell einfügen in CK2 :
=MAX(BH2:CJ2)")
MsgBox ("BH2 bis CK2 markieren und runterkopieren bis zu letzten Zeile")
End Sub

```

```

Sub StartAll_ChangeRequests()
    Dim wbdate As String
    wbdate = "2012_10_16"
    Call m_Main_ChangeRequests1(wbdate)
    Call m_Main_ChangeRequests2(wbdate)
    Call m_Main_ChangeRequests3(wbdate)
    Call m_Main_ChangeRequests4(wbdate)
    Call m_Main_ChangeRequests5(wbdate)
End Sub

Sub m_Main_ChangeRequests1(Optional wbdate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_RedFlags", "Begin", "task: EAC comparison", "file: EAC Comparison +
Template_RedFlags", CStr(Now()), ""))
    *****

    Dim flag As Integer
    'Dim wbdate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    If wbdate = Empty Then
        wbdate = "2012_10_16" 'VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) "'2012_2_8"
    End If
    reportversion_folder = "_V1-0" 'output of a02_Main_GenerateReport
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String
    Dim Wb_GenPart2 As Workbook
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part2_" & wbdate &
reportversion_file & ".xlsb"
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart2)
    End If
    If flag Then
        Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_8", Wb_GenPart2, Wb_GenPart2,
"Begin")
        Sheets("PMEC_VBA_MasterDataSet_8").name = "PMEC_VBA_MasterDataSet_8_bCR"
        Sheets("PMEC_VBA_MasterDataSet_8 (2)").name = "PMEC_VBA_MasterDataSet_8"
    End If
    *****
    *****

```

Dim Pild\_Col As Long

\*\*\*\*\*  
\*\*\*\*\*

'change PI Customer and Task Customer = Global Supply into Production & Supply  
'not used until november 2012 since TK Customer not migrated yet, PI Customer not important  
'do not forget to change the portfolio filtering rules for TK Customer!!!!

'If flag Then

```
' MsgBox ("Customer")
' Stop
' Col_PiCustomer = z_GetColumnIndex("PI Customer", 1, "PMEC_VBA_MasterDataSet_8")
' Col_TkCustomer = z_GetColumnIndex("Task Customer", 1, "PMEC_VBA_MasterDataSet_8")
' Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_8")
' RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_8")
' For Row = 2 To RowSize
'     If Cells(Row, Col_PiCustomer) = "GLOBAL SUPPLY" Then
'         Cells(Row, Col_PiCustomer) = "PRODUCTION & SUPPLY"
'     End If
'     If Cells(Row, Col_TkCustomer) = "GLOBAL SUPPLY" Then
'         Cells(Row, Col_TkCustomer) = "PRODUCTION & SUPPLY"
'     End If
' Next Row
'End If
```

\*\*\*\*\*  
\*\*\*\*\*

If flag Then

```
MsgBox ("Status change to include or exclude PIs")
Stop
col_PiStatus = z_GetColumnIndex("PI Status", 1, "PMEC_VBA_MasterDataSet_8")
Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_8")
RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_8")
For Row = 2 To RowSize
    Cells(Row, col_PiStatus).Activate
    If Cells(Row, Pild_Col) = "PI0010224" Then
        Cells(Row, col_PiStatus) = "Evaluation"
        Cells(Row, col_PiStatus).Font.Color = RGB(0, 255, 0)
    End If
    If Cells(Row, Pild_Col) = "PI0009266" Then
        Cells(Row, col_PiStatus) = "Planned"
        Cells(Row, col_PiStatus).Font.Color = RGB(0, 255, 0)
    End If
    If Cells(Row, Pild_Col) = "PI0009837" Then
        Cells(Row, col_PiStatus) = "Planned"
        Cells(Row, col_PiStatus).Font.Color = RGB(0, 255, 0)
    End If
Next
Call z_WorkbookSave(Wb_GenPart2)
End If
```

\*\*\*\*\*  
\*\*\*\*\*

'change Program names (already changed in SmC, therefore this code did nothing)

If flag Then

```

MsgBox ("Syngenta Program name changes")
Stop
Col_SynProg = z_GetColumnIndex("PI Syngenta Program", 1, "PMEC_VBA_MasterDataSet_8")
Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_8")
RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_8")
For Row = 2 To RowSize
    If Cells(Row, Col_SynProg) = "LOW RES ICM VEG SPEC CROP" Then
        Cells(Row, Col_SynProg) = "SpC MS gain through ICM"
        Cells(Row, Col_SynProg).Font.Color = RGB(0, 255, 0)
    End If
    If Cells(Row, Col_SynProg) = "Potato Y&Q CN" Then
        Cells(Row, Col_SynProg) = "SpC Potato Healthy Tubers"
        Cells(Row, Col_SynProg).Font.Color = RGB(0, 255, 0)
    End If
Next Row
Call z_WorkbookSave(Wb_GenPart2)
End If

```

```

*****
*****

```

```

'Delete Resources
If flag Then
    MsgBox ("Delete resources")
    Stop
    'still old name in sheet 8
    'Col_ResourceId = z_GetColumnIndex("RsIdentifier", 1, "PMEC_VBA_MasterDataSet_8")
    Col_ResourceId = z_GetColumnIndex("MyResourceId", 1, "PMEC_VBA_MasterDataSet_8")
    Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_8")
    RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_8")
    ColSize = z_ColSize(1, "PMEC_VBA_MasterDataSet_8")
    For Row = 2 To RowSize
        If Cells(Row, Col_ResourceId) = "RS00090851" Then
            Cells(Row, Col_ResourceId).Activate
            For col = 2 To ColSize
                If Cells(1, col) Like "EAC*" Then
                    Cells(Row, col).Activate
                    If Cells(Row, col) <> 0 Then
                        Cells(Row, col) = 0
                        Cells(Row, col).Font.Color = RGB(0, 255, 0)
                    End If
                End If
            Next
        End If
    Next
End If
Next Row
Call z_WorkbookSave(Wb_GenPart2)
End If

```

```

*****
*****

```

```

'change PI Sub Type (already done, code not necessary)
If flag Then
    MsgBox ("change pi sub type")
    Stop

```

```

Col_SubType = z_GetColumnIndex("PI Sub Type", 1, "PMEC_VBA_MasterDataSet_8")
Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_8")
RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_8")
For Row = 2 To RowSize
'   If Cells(Row, Pild_Col) = "PI0009725" Then
'       Cells(Row, Col_SubType) = "Non-Product Customer Offer"
'       Cells(Row, Col_SubType).Font.Color = RGB(0, 255, 0)
'   End If
'   If Cells(Row, Pild_Col) = "PI0010183" Then
'       Cells(Row, Col_SubType) = "Non-Product Customer Offer"
'       Cells(Row, Col_SubType).Font.Color = RGB(0, 255, 0)
'   End If
'   If Cells(Row, Pild_Col) = "PI0009494" Then
'       Cells(Row, Col_SubType).Activate
'       Cells(Row, Col_SubType) = "Platform"
'       Cells(Row, Col_SubType).Font.Color = RGB(0, 255, 0)
'   End If
Next Row
Call z_WorkbookSave(Wb_GenPart2)
End If

'*****
'*****

'change List of strategic crops
If flag Then
    MsgBox ("change list of strategic crops")
    Stop
    Col_ListOfStrategicCrops = z_GetColumnIndex("PI List of Strategic Crops", 1,
"PMEC_VBA_MasterDataSet_8")
    Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_8")
    RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_8")
    For Row = 2 To RowSize
        If Cells(Row, Pild_Col) = "PI0002554" Then
            Cells(Row, Col_ListOfStrategicCrops).Activate
            Cells(Row, Col_ListOfStrategicCrops) = "Lawn & Garden (100 %)"
            Cells(Row, Col_ListOfStrategicCrops).Font.Color = RGB(0, 255, 0)
        End If
        If Cells(Row, Pild_Col) = "PI0002595" Then
            Cells(Row, Col_ListOfStrategicCrops).Activate
            Cells(Row, Col_ListOfStrategicCrops) = "Lawn & Garden (100 %)"
            Cells(Row, Col_ListOfStrategicCrops).Font.Color = RGB(0, 255, 0)
        End If
        If Cells(Row, Pild_Col) = "PI0002642" Then
            Cells(Row, Col_ListOfStrategicCrops).Activate
            Cells(Row, Col_ListOfStrategicCrops) = "Lawn & Garden (100 %)"
            Cells(Row, Col_ListOfStrategicCrops).Font.Color = RGB(0, 255, 0)
        End If
        If Cells(Row, Pild_Col) = "PI0004182" Then
            Cells(Row, Col_ListOfStrategicCrops).Activate
            Cells(Row, Col_ListOfStrategicCrops) = "Lawn & Garden (100 %)"
            Cells(Row, Col_ListOfStrategicCrops).Font.Color = RGB(0, 255, 0)
        End If
        If Cells(Row, Pild_Col) = "PI0009313" Then

```

```

        Cells(Row, Col_ListOfStrategicCrops).Activate
        Cells(Row, Col_ListOfStrategicCrops) = "Lawn & Garden (100 %)"
        Cells(Row, Col_ListOfStrategicCrops).Font.Color = RGB(0, 255, 0)
    End If
    If Cells(Row, Pild_Col) = "PI0009314" Then
        Cells(Row, Col_ListOfStrategicCrops).Activate
        Cells(Row, Col_ListOfStrategicCrops) = "Lawn & Garden (100 %)"
        Cells(Row, Col_ListOfStrategicCrops).Font.Color = RGB(0, 255, 0)
    End If
Next Row
Call z_WorkbookSave(Wb_GenPart2)
End If
*****
*****

'change List of strategic crops
If flag Then
    MsgBox ("change list of strategic crops: List compiled 5.9 and downloaded 7.9")
    Stop
End If

*****
*****

End Sub

Sub m_Main_ChangeRequests2(Optional wbddate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_RedFlags", "Begin", "task: EAC comparison", "file: EAC Comparison +
Template_RedFlags", CStr(Now()), ""))
    *****

    Dim flag As Integer
    'Dim wbddate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    If wbddate = Empty Then
        wbddate = "2012_10_16" 'VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) "'2012_2_8"
    End If
    reportversion_folder = "_V1-0" 'output of a02_Main_GenerateReport
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String
    Dim Wb_GenPart4 As Workbook
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbddate &
reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part4_" & wbddate &
reportversion_file & ".xlsb"
    End If
End Sub

```

```

    Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart4)
End If
If flag Then
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart4,
Wb_GenPart4, "Begin")
    Sheets("PMEC_VBA_MasterDataSet_13").name = "PMEC_VBA_MasterDataSet_13_bCR"
    Sheets("PMEC_VBA_MasterDataSet_13 (2)").name = "PMEC_VBA_MasterDataSet_13"
End If
*****
*****

Dim Pild_Col As Long
*****
*****

*****
*****

'change for RBS1=ProductSafety TkCustomer=M&S to CPD (no rbs in sheet 8 do it in sheet 13!)
If flag Then
    MsgBox ("TkCustomer change")
    Stop
    Col_TkCustomer = z_GetColumnIndex("TK Task Customer", 1, "PMEC_VBA_MasterDataSet_13")
    Col_RBS1 = z_GetColumnIndex("RBS Level 1", 1, "PMEC_VBA_MasterDataSet_13")
    Pild_Col = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_13")
    RowSize = z_RowSize(Pild_Col, "PMEC_VBA_MasterDataSet_13")
    For Row = 2 To RowSize
        If Cells(Row, Col_RBS1) = "Product Safety" Then
            Cells(Row, Col_RBS1).Activate
            If Cells(Row, Col_TkCustomer) = "MARKETING & SALES" Then
                Cells(Row, Col_TkCustomer).Activate
                Cells(Row, Col_TkCustomer) = "CP DEVELOPMENT"
                Cells(Row, Col_TkCustomer).Font.Color = RGB(0, 255, 0)
            End If
        End If
    Next Row
    Call z_WorkbookSave(Wb_GenPart4)
End If
*****
*****

*****
*****

*****
*****

End Sub

Sub m_Main_ChangeRequests3(Optional wbdate As String)
    If flag Then
        MsgBox ("change ICS attribute according to Jasper's list")
        Stop
        'Set everythingthing to NO

```

```

        'set the autofilter with the PIs on the list and set YES (or map it in)
    End If

End Sub

Sub start45789()
    Dim wbdate As String
    wbdate = "2012_10_16"
    Call m_Main_ChangeRequests4(wbdate)
End Sub

Sub m_Main_ChangeRequests4(Optional wbdate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_RedFlags", "Begin", "task: EAC comparison", "file: EAC Comparison +
Template_RedFlags", CStr(Now()), ""))
    *****

    Dim flag As Integer
    'Dim wbdate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    If wbdate = Empty Then
        wbdate = "2012_10_16" & VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) & "2012_2_8"
    End If
    reportversion_folder = "_V1-0" & "output of a02_Main_GenerateReport"
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String
    Dim Wb_PR As Workbook
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\CurrentVersionOnTeamspace_2012_09_19\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PortfolioReporting_2012_09_03"
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_PR)
    End If

    *****
    *****

    Dim Pild_Col As Long
    *****
    *****

    '1. Manually added in EACFull2012,EACFull,EACOther2012,EACDirect2012,EACDirect:
    ' PI10207GlobalSupportFunctions: $169'652'000
    ' PI10207RD-IS: $19'000'000

```



'2. change all 2012 numbers according to Ebru's table

If flag Then

MsgBox ("change 2012 numbers")

Stop

Dim WSh\_from As Worksheet

Dim WSh\_to As Worksheet

Set WSh\_from = Sheets("ChangeRequest2012\_Ebru")

Set WSh\_to = Sheets("R3 SmC\_MasterDataSet")

Dim Col\_InnovLifeCycle As Long

Dim Col\_InvestmentSegment As Long

Dim Col\_SubTypes As Long

Dim Col\_StrategicCrop As Long

Dim Col\_PL1 As Long

Dim Col\_EACFull2012 As Long

Dim Col\_EACFullTotal As Long

Dim Col\_Pild As Long

Col\_InnovLifeCycle = z\_GetColumnIndex("InnovationLifeCycle", 1, WSh\_to.name)

Col\_InvestmentSegment = z\_GetColumnIndex("InvestmentSegment", 1, WSh\_to.name)

Col\_SubTypes = z\_GetColumnIndex("PI Sub Type", 1, WSh\_to.name)

Col\_StrategicCrop = z\_GetColumnIndex("2012 PI Strategic Crop", 1, WSh\_to.name)

Col\_PL1 = z\_GetColumnIndex("Portfolio Level 1", 1, WSh\_to.name)

,

Col\_EACFull2012 = z\_GetColumnIndex("EAC Full Costs 2012", 1, WSh\_to.name)

Col\_EACFullTotal = z\_GetColumnIndex("EAC Full Costs", 1, WSh\_to.name)

Col\_EACSD2012 = z\_GetColumnIndex("EAC SD Full Costs 2012", 1, WSh\_to.name)

Col\_EACSDTotal = z\_GetColumnIndex("EAC SD Full Costs", 1, WSh\_to.name)

Col\_EACTrial2012 = z\_GetColumnIndex("EAC Trials Full Costs 2012", 1, WSh\_to.name)

Col\_EACTrialTotal = z\_GetColumnIndex("EAC Trials Full Costs", 1, WSh\_to.name)

Col\_EACOther2012 = z\_GetColumnIndex("EAC Other \$ 2012", 1, WSh\_to.name)

Col\_EACOtherTotal = z\_GetColumnIndex("EAC Other \$", 1, WSh\_to.name)

Col\_EACExt2012 = z\_GetColumnIndex("EAC Ext \$ 2012", 1, WSh\_to.name)

Col\_EACExtTotal = z\_GetColumnIndex("EAC Ext \$", 1, WSh\_to.name)

Col\_EACDirect2012 = z\_GetColumnIndex("EAC Direct Costs 2012", 1, WSh\_to.name)

Col\_EACDirectTotal = z\_GetColumnIndex("EAC Direct Costs", 1, WSh\_to.name)

,

Col\_Pild = z\_GetColumnIndex("Pildentifier", 1, WSh\_to.name)

Dim Dollar As Double

Dim Percent As Double

Dim Row\_StrategicCrop\_from As Long

Dim Col\_from As Long

Dim Col\_B\_from As Long

```

Dim Col_C_from As Long
Dim Col_D_from As Long
Dim Col_M_from As Long
Row_StrategicCrop_from = 2
Col_D_from = 4
Col_M_from = 13
Col_B_from = 2
Col_C_from = 3

```

```

RowSize = z_RowSize(Col_Pild, WSh_to.name)
For Row = 2 To RowSize
    'Read StrategicCrop
    If WSh_to.Cells(Row, Col_StrategicCrop) = "Cereals" Then
        Col_from = 4
    ElseIf WSh_to.Cells(Row, Col_StrategicCrop) = "Corn" Then
        Col_from = 5
    ElseIf WSh_to.Cells(Row, Col_StrategicCrop) = "DFC" Then
        Col_from = 6
    ElseIf WSh_to.Cells(Row, Col_StrategicCrop) = "Lawn & Garden" Then
        Col_from = 7
    ElseIf WSh_to.Cells(Row, Col_StrategicCrop) = "Non-Crop" Then
        Col_from = 8
    ElseIf WSh_to.Cells(Row, Col_StrategicCrop) = "Rice" Then
        Col_from = 9
    ElseIf WSh_to.Cells(Row, Col_StrategicCrop) = "Soybean" Then
        Col_from = 10
    ElseIf WSh_to.Cells(Row, Col_StrategicCrop) = "Speciality" Then
        Col_from = 11
    ElseIf WSh_to.Cells(Row, Col_StrategicCrop) = "Sugarcane" Then
        Col_from = 12
    ElseIf WSh_to.Cells(Row, Col_StrategicCrop) = "Vegetables" Then
        Col_from = 13
    Else
        Stop
    End If

```

```

'get percent
Percent = 0
'If Not WSh_to.Cells(Row, Col_SubTypes) Like "Capability*" Then
'Chemicals
'*****
If WSh_to.Cells(Row, Col_InnovLifeCycle) = "3.1.Chemicals - Research Incl. PER" Then
    Percent = WSh_from.Cells(3, Col_from)
End If
If WSh_to.Cells(Row, Col_InnovLifeCycle) = "3.2.Chemicals - New AI Development" Then
    Percent = WSh_from.Cells(4, Col_from)
End If
If WSh_to.Cells(Row, Col_InnovLifeCycle) = "3.3.Chemicals - New Products & Extensions" Then
    Percent = WSh_from.Cells(5, Col_from)
End If
If WSh_to.Cells(Row, Col_InnovLifeCycle) = "3.4.Chemicals - Product Maintenance" Then
    Percent = WSh_from.Cells(6, Col_from)
End If

```

```

'Genetics
*****

If WSh_to.Cells(Row, Col_InvestmentSegment) = "1.a.Genetic Modification Trait" Then
    Percent = WSh_from.Cells(13, Col_from)
End If
If WSh_to.Cells(Row, Col_InvestmentSegment) = "1.c.Breeding" Then
    Percent = WSh_from.Cells(14, Col_from)
End If
If WSh_to.Cells(Row, Col_InvestmentSegment) = "1.b.Native Traits" Then
    Percent = WSh_from.Cells(15, Col_from)
End If
'End If
'Capability*
*****

If WSh_to.Cells(Row, Col_SubTypes) Like "Capability*" Then
    If Not WSh_to.Cells(Row, Col_PL1) = "CROP_PROTECTION" Then
        Percent = WSh_from.Cells(17, Col_from)
    End If
    If Not WSh_to.Cells(Row, Col_PL1) = "SEEDS" Then
        Percent = WSh_from.Cells(18, Col_from)
    End If
End If
'NewTechnology
*****

If WSh_to.Cells(Row, Col_InvestmentSegment) = "2.a.Integrated Solutions" Then
    Percent = WSh_from.Cells(8, Col_from)
End If
If WSh_to.Cells(Row, Col_InvestmentSegment) = "2.c.Bio Controls" Then
    Percent = WSh_from.Cells(9, Col_from)
End If
If WSh_to.Cells(Row, Col_InvestmentSegment) = "2.d.Crop Enhancement" Then
    Percent = WSh_from.Cells(10, Col_from)
End If
If WSh_to.Cells(Row, Col_InvestmentSegment) = "2.b.Adjacent Technology" Then
    Percent = WSh_from.Cells(11, Col_from)
End If

'Platform*
*****

If WSh_to.Cells(Row, Col_InnovLifeCycle) = "4.3.Global Functions & Capabilities - Crop Specific
Platforms" Then
    If WSh_to.Cells(Row, Col_PL1) <> "SEEDS" Then
        Percent = WSh_from.Cells(20, Col_from)
    End If
    If WSh_to.Cells(Row, Col_PL1) = "SEEDS" Then
        Percent = WSh_from.Cells(19, Col_from)
    End If
End If
If WSh_to.Cells(Row, Col_InnovLifeCycle) = "4.4.Global Functions & Capabilities - Cross Crop
Platforms" Then
    If WSh_to.Cells(Row, Col_PL1) <> "SEEDS" Then
        Percent = WSh_from.Cells(20, Col_from)
    End If

```

```

If WSh_to.Cells(Row, Col_PL1) = "SEEDS" Then
    Percent = WSh_from.Cells(19, Col_from)
End If
End If

```

```

If Percent <> 0 Then

```

```

    'Change EAC

```

```

    *****

```

```

    Dollar = WSh_to.Cells(Row, Col_EACFull2012) * Percent

```

```

    If Dollar <> 0 Then

```

```

        WSh_to.Cells(Row, Col_EACFull2012) = WSh_to.Cells(Row, Col_EACFull2012) + Dollar

```

```

        WSh_to.Cells(Row, Col_EACFullTotal) = WSh_to.Cells(Row, Col_EACFullTotal) + Dollar

```

```

        WSh_to.Cells(Row, Col_EACFullTotal).Font.Color = RGB(255, 0, 0)

```

```

    ,

```

```

        Dollar = WSh_to.Cells(Row, Col_EACSD2012) * Percent

```

```

        If Dollar <> 0 Then

```

```

            WSh_to.Cells(Row, Col_EACSD2012) = WSh_to.Cells(Row, Col_EACSD2012) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACSDTotal) = WSh_to.Cells(Row, Col_EACSDTotal) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACSDTotal).Font.Color = RGB(255, 0, 0)

```

```

        End If

```

```

    ,

```

```

        Dollar = WSh_to.Cells(Row, Col_EACTrial2012) * Percent

```

```

        If Dollar <> 0 Then

```

```

            WSh_to.Cells(Row, Col_EACTrial2012) = WSh_to.Cells(Row, Col_EACTrial2012) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACTrialTotal) = WSh_to.Cells(Row, Col_EACTrialTotal) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACTrialTotal).Font.Color = RGB(255, 0, 0)

```

```

        End If

```

```

    ,

```

```

        Dollar = WSh_to.Cells(Row, Col_EACOther2012) * Percent

```

```

        If Dollar <> 0 Then

```

```

            WSh_to.Cells(Row, Col_EACOther2012) = WSh_to.Cells(Row, Col_EACOther2012) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACOtherTotal) = WSh_to.Cells(Row, Col_EACOtherTotal) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACOtherTotal).Font.Color = RGB(255, 0, 0)

```

```

        End If

```

```

    ,

```

```

        Dollar = WSh_to.Cells(Row, Col_EACExt2012) * Percent

```

```

        If Dollar <> 0 Then

```

```

            WSh_to.Cells(Row, Col_EACExt2012) = WSh_to.Cells(Row, Col_EACExt2012) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACExtTotal) = WSh_to.Cells(Row, Col_EACExtTotal) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACExtTotal).Font.Color = RGB(255, 0, 0)

```

```

        End If

```

```

    ,

```

```

        Dollar = WSh_to.Cells(Row, Col_EACDirect2012) * Percent

```

```

        If Dollar <> 0 Then

```

```

            WSh_to.Cells(Row, Col_EACDirect2012) = WSh_to.Cells(Row, Col_EACDirect2012) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACDirectTotal) = WSh_to.Cells(Row, Col_EACDirectTotal) + Dollar

```

```

            WSh_to.Cells(Row, Col_EACDirectTotal).Font.Color = RGB(255, 0, 0)

```

```

        End If

```

```

    End If

```

```

End If

```

```

Next Row

```

```
Call z_WorkbookSave(Wb_PR)
End If
```

```
*****
*****
```

```
End Sub
```

```
Sub m_Main_ChangeRequests5(Optional wbdate As String)
```

```
    If flag Then
```

```
        MsgBox ("change in portfolio reporting data set  
BBandSP_changerequest_vegetables_2012_09_24")
```

```
        Stop
```

```
    End If
```

```
    flag = 1
```

```
    If flag Then
```

```
        MsgBox ("change portfolio reporting data set the BB terminology for Soybean")
```

```
        Stop
```

```
        Col_SC = z_GetColumnIndex("2012 PI Strategic Crop", 1, "R3 SmC_MasterDataSet")
```

```
        Col_BB = z_GetColumnIndex("PI Building Blocks", 1, "R3 SmC_MasterDataSet")
```

```
        Dim Col_Pild As Long
```

```
        Col_Pild = z_GetColumnIndex("Pildentifier", 1, "R3 SmC_MasterDataSet")
```

```
        RowSize = z_RowSize(Col_Pild, "R3 SmC_MasterDataSet")
```

```
        For Row = 2 To RowSize
```

```
            If Cells(Row, Col_SC) = "Soybean" Then
```

```
                Cells(Row, Col_SC).Activate
```

```
                If LCase(Cells(Row, Col_BB)) = "soy - building block 1" Or LCase(Cells(Row, Col_BB)) = "soy -  
building block 1 " Then
```

```
                    Cells(Row, Col_BB).Activate
```

```
                    Cells(Row, Col_BB) = "Weed management"
```

```
                    Cells(Row, Col_BB).Font.Color = RGB(0, 255, 0)
```

```
                ElseIf LCase(Cells(Row, Col_BB)) = "soy - building block 2" Then
```

```
                    Cells(Row, Col_BB).Activate
```

```
                    Cells(Row, Col_BB) = "Insect control"
```

```
                    Cells(Row, Col_BB).Font.Color = RGB(0, 255, 0)
```

```
                ElseIf LCase(Cells(Row, Col_BB)) = "soy - building block 3" Then
```

```
                    Cells(Row, Col_BB).Activate
```

```
                    Cells(Row, Col_BB) = "Disease control"
```

```
                    Cells(Row, Col_BB).Font.Color = RGB(0, 255, 0)
```

```
                ElseIf LCase(Cells(Row, Col_BB)) = "soy - building block 4" Then
```

```
                    Cells(Row, Col_BB).Activate
```

```
                    Cells(Row, Col_BB) = "Nematode control"
```

```
                    Cells(Row, Col_BB).Font.Color = RGB(0, 255, 0)
```

```
                ElseIf LCase(Cells(Row, Col_BB)) = "soy - building block 5" Then
```

```
                    Cells(Row, Col_BB).Activate
```

```
                    Cells(Row, Col_BB) = "Increase Seeds footprint"
```

```
                    Cells(Row, Col_BB).Font.Color = RGB(0, 255, 0)
```

```
                ElseIf LCase(Cells(Row, Col_BB)) = "soy - building block 6" Then
```

```
                    Cells(Row, Col_BB).Activate
```

```
                    Cells(Row, Col_BB) = "Yield step change"
```

```
                    Cells(Row, Col_BB).Font.Color = RGB(0, 255, 0)
```

```
                ElseIf LCase(Cells(Row, Col_BB)) = "soy - building block 7" Then
```

```
                    Cells(Row, Col_BB).Activate
```

```
                    Cells(Row, Col_BB) = "Drought stability"
```

```

        Cells(Row, Col_BB).Font.Color = RGB(0, 255, 0)
    ElseIf LCase(Cells(Row, Col_BB)) = "soy - building block 8" Then
        Cells(Row, Col_BB).Activate
        Cells(Row, Col_BB) = "Land optimization"
        Cells(Row, Col_BB).Font.Color = RGB(0, 255, 0)
    ElseIf LCase(Cells(Row, Col_BB)) = "soy - building block 9" Then
        Cells(Row, Col_BB).Activate
        Cells(Row, Col_BB) = "Value chain solutions"
        Cells(Row, Col_BB).Font.Color = RGB(0, 255, 0)
    Else
        Stop
    End If
End If
Next Row
End If

End Sub

Sub StartAll_RedFlags()
    Dim wbdate As String
    wbdate = "2012_10_16"
    Call m_Main_RedFlags(wbdate)
End Sub

Sub m_Main_RedFlags(Optional wbdate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_RedFlags", "Begin", "task: EAC comparison", "file: EAC Comparison +
Template_RedFlags", CStr(Now()), ""))
    *****

    Dim flag As Integer
    'Dim wbdate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    If wbdate = Empty Then
        wbdate = "2012_10_16" 'VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) "'2012_2_8"
    End If
    reportversion_folder = "_V1-0" 'output of a02_Main_GenerateReport
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String
    Dim Wb_GenPart1 As Workbook
    Dim Wb_GenPart2 As Workbook
    Dim Wb_GenPart4 As Workbook
    Dim Wb_TemplateRedFlags As Workbook
    If flag Then

```

```

    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part4_" & wbdate &
reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart4)
End If
If flag Then
    Call z_ShNew("EAC_Comparison", "Begin")
End If
    'copy a sheet from another workbook
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part1_" & wbdate &
reportversion_file & ".xlsb"
    'Open Workbook may fail if it needs to be repaired (remove sort from sheet6)
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart1)
End If
If flag Then
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_PiDataSet_2", Wb_GenPart1, Wb_GenPart4,
"Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_GenPart1.Close False
End If
    'copy a sheet from another workbook
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
    WbName = "Template_RedFlags.xlsb"
    'Open Workbook
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_TemplateRedFlags)
End If
If flag Then
    Call z_CopySheetFromWb1ToWb2("Template_RedFlags", Wb_TemplateRedFlags, Wb_GenPart4,
"Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_TemplateRedFlags.Close False
End If
    *****

    '11 minutes
    Call z_Create_EacComparison
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_RedFlags", "End", "task: EAC comparison", "file: EAC Comparison +
Template_RedFlags", CStr(Now()), ""))
    *****

If flag Then
    Call z_WorkbookSave(Wb_GenPart4)
End If
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part2_" & wbdate &
reportversion_file & ".xlsb"

```

```

'Open Workbook may fail if it needs to be repaired (remove sort from sheet6)
Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart2)
End If
If flag Then
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_9", Wb_GenPart2, Wb_GenPart4,
    "Begin")
    'Nof Rows: Crop split
    Sheets("Template_RedFlags").Range("B14").Value = z_RowSize(1, "PMEC_VBA_MasterDataSet_9")
    'remove sheet again because of memory problems
    Call z_ShDelete("PMEC_VBA_MasterDataSet_9", Wb_GenPart4)
    'close Wb_GenPart1, closes the active workbook and saves any changes

    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_7", Wb_GenPart2, Wb_GenPart4,
    "Begin")
    'Nof Rows: Resources
    Sheets("Template_RedFlags").Range("B12").Value = z_RowSize(1, "PMEC_VBA_MasterDataSet_7")

    Wb_GenPart2.Close False
End If
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
    reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part1_" & wbdate &
    reportversion_file & ".xlsb"
    'Open Workbook may fail if it needs to be repaired (remove sort from sheet6)
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart1)
End If
If flag Then
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_1", Wb_GenPart1, Wb_GenPart4,
    "Begin")
    Call z_CopySheetFromWb1ToWb2("Log_3", Wb_GenPart1, Wb_GenPart4, "Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_GenPart1.Close False
End If
    'Nof Rows: Downloaded lines
    Sheets("Template_RedFlags").Range("B10").Value = z_RowSize(1, "PMEC_VBA_MasterDataSet_1")
    'Nof Rows: Removed Titles
    Sheets("Template_RedFlags").Range("B11").Value = z_RowSize(5, "Log_3")
If flag Then
    'remove sheet again because of memory problems
    Call z_ShDelete("PMEC_VBA_MasterDataSet_1", Wb_GenPart4)
    Call z_ShDelete("Log_3", Wb_GenPart4)
End If
    *****

    Call z_CalculateAndCompile_RedFlags
    *****

If flag Then
    'remove sheet again because of memory problems
    Call z_ShDelete("PMEC_VBA_MasterDataSet_7", Wb_GenPart4)
    Call z_ShDelete("PMEC_VBA_PiDataSet_2", Wb_GenPart4)
End If
If flag Then
    Call z_WorkbookSave(Wb_GenPart4)

```



```

End If
'Copy the sheet into the summary workbook
Dim Wb_OutputChkOnWholeReport As Workbook
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Outputs_MainGenerateReport\"
WbName = "OutputChecksOn_WholeReports.xlsm"
'*****
'*****

If flag Then
    Call z_ExcelSessionWindowMinimized(Wb_GenPart4)
    Call z_ExcelSessionWindowNormal(Wb_GenPart4)
    MsgBox ("Check the sheet RedFlags")
    Stop
End If
'open
If flag Then
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_OutputChkOnWholeReport)
End If
'move
If flag Then
    Call z_CopySheetFromWb1ToWb2("Template_RedFlags", Wb_GenPart4,
Wb_OutputChkOnWholeReport, "Begin")
End If
'rename
If flag Then
    Wb_OutputChkOnWholeReport.Sheets("Template_RedFlags").name = wdate
End If
If flag Then
    'close Wb_RBS, closes the active workbook and saves any changes
    Wb_OutputChkOnWholeReport.Close True
End If
'*****
'*****

Dim Wb_OutputTimeseriesOnEac As Workbook
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Outputs_MainGenerateReport\"
WbName = "OutputTimeseriesOn_EacAndDeltaEac.xlsm"
'*****
'*****

'open
If flag Then
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_OutputTimeseriesOnEac)
End If
If flag Then
    'EAC Total Full Costs
    Dim RowSize As Long
    Dim FirstYr As String
    Dim FirstCost As String
    Dim FirstCost_Col As Long
    Wb_GenPart4.Activate
    RowSize = z_RowSize(1, "PMEC_VBA_MasterDataSet_13")
    FirstYr = year(Now()) - 1
    FirstCost = "EAC # SD " & FirstYr & "_c"

```

```

FirstCost_Col = z_GetColumnIndex(FirstCost, 1, "PMEC_VBA_MasterDataSet_13")
LastCost_Col = FirstCost_Col + 71
Dim TotalSum As Double
Dim Col_to As Long
Wb_OutputTimeseriesOnEac.Activate
Col_to = z_ColSize(1, "Summary_EACs") + 1
Dim Row_to As Long
Row_to = 2
For col = FirstCost_Col To LastCost_Col
    Wb_GenPart4.Activate
    Sheets("PMEC_VBA_MasterDataSet_13").Select
    TotalSum = WorksheetFunction.Sum(Range(Cells(2, col), Cells(RowSize, col)))
    Wb_OutputTimeseriesOnEac.Activate
    Sheets("Summary_EACs").Cells(Row_to, Col_to).Value = TotalSum
    Row_to = Row_to + 1
Next col
End If
If flag Then
    'Add Titles
    Cells(1, Col_to).Value = wbdate
End If
If flag Then
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_RedFlags", "End", "task:", "file: EAC Comparison + Template_RedFlags",
CStr(Now()), ""))
    Call z_ExcelSessionWindowMinimized(Wb_GenPart4)
    Call z_ExcelSessionWindowNormal(Wb_GenPart4)
    MsgBox ("Check the sheet OutputTimeseriesOn_EacAndDeltaEac")
    Stop
End If
End Sub

```

```

Sub StartAll_PivotFlat()
    Dim wbdate As String
    wbdate = "2012_10_16"
    Call m_Main_PivotFlat_PiAndTK_CustomizedDataSet_FunctionalReporting(wbdate)
    Call m_Main_PivotFlat_PiAndTK_CustomizedForPortfolioReporting(wbdate)
    Call m_Main_MemoryLeakWorkaround_10
End Sub
Sub StartAll_PivotFlat2()
    Dim wbdate As String
    wbdate = "2012_10_16"
    Call m_Main_PivotFlat_PiAndTK_CustomizedDataSet_OneLinePerId(wbdate)
    Call m_Main_PivotFlat_Rs_CustomizedDataSet_OneLinePerId(wbdate)
End Sub

```

\*\*\*\*\*

'Before starting these tasks:  
 '1. Close Excel completely!

'2. Close Outlook!

'3. Restart Excel

\*\*\*\*\*

Sub m\_Main\_PivotFlat\_PiAndTK\_CustomizedDataSet\_FunctionalReporting(Optional wdate As String)

Call z\_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", \_  
Array("Main\_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))

\*\*\*\*\*

Dim flag As Integer

Dim flag\_Tk As Integer

'Dim wdate As String

Dim reportversion\_folder As String

Dim reportversion\_file As String

\*\*\*\*\*

'reopen

flag = 1

flag\_Tk = 0 'default=0

If wdate = Empty Then

wdate = "2012\_10\_16" 'VBA.DateTime.Year(Now()) & "\_" & VBA.DateTime.Month(Now()) & "\_"  
& VBA.DateTime.Day(Now()) "2012\_2\_8"

End If

reportversion\_folder = "\_V1-0"

reportversion\_file = "\_V1-0"

folderdescription = ""

\*\*\*\*\*

'Open and activate an Excel workbook (and session)

Dim WbPath As String

Dim WbName As String

Dim Wb\_GenPart4 As Workbook

Dim wb\_new As Workbook

\*\*\*\*\*

\*\*\*\*\*

'Adds the new workbook RD\_MasterDataSet

\*\*\*\*\*

\*\*\*\*\*

'Add a new workbook and move the MasterDataSet and the PiDataSet into it

If flag Then

WbPath = "C:\Users\t740698\Desktop\PMEC\_VBA\_Master-DataSet\" & wdate &  
reportversion\_folder & folderdescription & "\"

WbName = "CONFIDENTIAL\_SmC\_PMEC\_VBA\_MasterDataSet\_PivotFlat\_ForFunctionalReporting\_"  
& wdate & reportversion\_file & ".xlsb" 'for test purposes

Call z\_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb\_new)

End If

'Open existing workbook

If flag Then

WbPath = "C:\Users\t740698\Desktop\PMEC\_VBA\_Master-DataSet\" & wdate &  
reportversion\_folder & folderdescription & "\"

WbName = "CONFIDENTIAL\_SmC\_PMEC\_VBA\_MasterDataSet\_Generation\_Part4\_" & wdate &  
reportversion\_file & ".xlsb"

Call z\_OpenAndActivateWb(WbName, WbPath, Wb\_GenPart4)

End If

'copy a sheet from another workbook

If flag Then

```

    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart4, wb_new,
"Begin")
'close Wb_GenPart1, closes the active workbook and saves any changes
'rename sheet
Sheets("PMEC_VBA_MasterDataSet_13").name = "SmC_MasterDataSet_ResourceLevel"
Wb_GenPart4.Close False
End If
If flag Then
'in case they have already been deleted in a previous run
On Error Resume Next
    Call z_ShDelete("Sheet1", wb_new)
    Call z_ShDelete("Sheet2", wb_new)
    Call z_ShDelete("Sheet3", wb_new)
On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If

'only keep needed columns from base data set (delete not needed columns)
If flag Then
    Dim ColumnNames_All As Variant
    ColumnNames_All = z_FillArray("SmC_MasterDataSet_ResourceLevel", , 1, 1)
    '
    Dim Wb_from As Workbook
    Dim Rng_from As Range
    Dim ColumnNames_ToKeep As Variant
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_PI_TK_RS", 17,
Wb_from)
    ColumnNames_ToKeep = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_PI_TK_RS",
Rng_from, "ReadACol", Wb_from)
    '
    Call z_DeleteColumns3("SmC_MasterDataSet_ResourceLevel", ColumnNames_ToKeep,
ColumnNames_All)
End If

If flag Then
'Stamp date
Dim StampDate As String
StampDate = z_StampDate(wbdate)
Dim FirstYr As Integer
FirstYr = VBA.DateTimes.year(Now()) - 1
Call z_AddPivotTable_LightVersion("SmC_MasterDataSet_ResourceLevel",
"PivotTable_ResourceLevel", "PiIdentifier", "Pi Title", "EAC Full Costs " & CStr(FirstYr + 1) & "_c")
Call z_AddTimeStamp("SmC_MasterDataSet_ResourceLevel", StampDate, 1, 1, 25)
Call z_AddTimeStamp("PivotTable_ResourceLevel", StampDate, 1, 1, 25)
Debug.Print Now()
End If

Call z_WorkbookSave(wb_new)
wb_new.Close
Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
    Array("Main_CheckBusinessRules", "End", "task:", "file:", CStr(Now()), ""))
End Sub

```

```

Sub m_Main_PivotFlat_PiAndTK_CustomizedForPortfolioReporting(Optional wbddate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
    *****

    Dim flag As Integer
    Dim flag_Tk As Integer
    'Dim wbddate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    flag_Tk = 0 'default=0
    If wbddate = Empty Then
        wbddate = "2012_10_16" 'VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) "'2012_2_8"
    End If
    reportversion_folder = "_V1-0"
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String
    Dim Wb_GenPart4 As Workbook
    Dim wb_new As Workbook
    *****
    *****

    'Adds the new workbook RD_MasterDataSet
    *****
    *****

    'Add a new workbook and move the MasterDataSet and the PiDataSet into it
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbddate &
reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForPortfolioReporting_"
& wbddate & reportversion_file & ".xlsb" 'for test purposes
        Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
    End If

    'Open existing workbook
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbddate &
reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part4_" & wbddate &
reportversion_file & ".xlsb"
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart4)
    End If

    'copy a sheet from another workbook
    If flag Then
        Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart4, wb_new,
"Begin")
        Sheets("PMEC_VBA_MasterDataSet_13").name = "PMEC_VBA_MasterDataSet_13_PI"
    End If
End Sub

```

```

    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart4, wb_new,
"Begin")
    Wb_GenPart4.Close False
End If
If flag Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", wb_new)
        Call z_ShDelete("Sheet2", wb_new)
        Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If
*****

'one line per PIIId@PiStrategicCrop@TkStatus@TkCustomer@RBS1@RBS2@RBS3
*****

'only keep needed columns from base data set (delete not needed columns)
'-----

If flag Then
    Dim ColumnNames_All As Variant
    ColumnNames_All = z_FillArray("PMEC_VBA_MasterDataSet_13_Pi", , 1, 1)
    ,

    Dim Wb_from As Workbook
    Dim Rng_from As Range
    Dim ColumnNames_ToKeep As Variant
    Dim ColumnNames_ToKeep_1 As Variant
    Dim ColumnNames_ToKeep_2 As Variant
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_Pi_TK_RS", 2,
Wb_from)
    ColumnNames_ToKeep_1 = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_Pi_TK_RS",
Rng_from, "ReadACol", Wb_from)
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_Pi_TK_RS", 3,
Wb_from)
    ColumnNames_ToKeep_2 = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_Pi_TK_RS",
Rng_from, "ReadACol", Wb_from)
    ColumnNames_ToKeep = z_Arrays_PutTogether(ColumnNames_ToKeep_1,
ColumnNames_ToKeep_2)
    ,

    Call z_DeleteColumns3("PMEC_VBA_MasterDataSet_13_Pi", ColumnNames_ToKeep,
ColumnNames_All)
End If
'aggregation via Pivot
'-----

If flag Then
    'Stampdate
    Dim StampDate As String
    StampDate = z_StampDate(wbdate)
    'Generate Pivot and flat table on PI level
    Debug.Print Now()
    PivotName_PIs =
z_CreatePivot_PIs_CustomisedForPortfolioReporting("PMEC_VBA_MasterDataSet_13_Pi",
"Pivot_PIs", "SmC_MasterDataSet_PiLevel")
    'remove the grand total line

```

```

    Call z_RemRow_WithAttrEntry("SmC_MasterDataSet_PiLevel", "Pildentifier", "Pildentifier", "Grand
Total", 0)
End If
'remove no longer needed sheets
'-----
If flag Then
    Call z_ShDelete("PMEC_VBA_MasterDataSet_13_Pi", wb_new)
    Call z_ShDelete("Pivot_PIs", wb_new)
End If
'mapping
'-----
If flag Then
    'read attributes
    Dim ColumnNames_ToMap As Variant
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_Pi_TK_RS", 4,
Wb_from)
    ColumnNames_ToMap = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_Pi_TK_RS",
Rng_from, "ReadACol", Wb_from)
    'insert attributes before Name_to/col_to
    Dim Name_to As String
    Name_to = ColumnNames_ToKeep_2(0)
    Dim Col_to As Long
    Col_to = z_GetColumnIndex(Name_to, 1, "SmC_MasterDataSet_PiLevel")
    Dim NofCols As Integer
    NofCols = UBound(ColumnNames_ToMap) - LBound(ColumnNames_ToMap) + 1
    Call z_InsertEmptyCols("SmC_MasterDataSet_PiLevel", NofCols, Col_to)
    Call z_AddColNames(ColumnNames_ToMap, "SmC_MasterDataSet_PiLevel", 1, Col_to)
    'add the concatenated key in Sheet to
(Pild@PiStrategicCrop@TkStatus@TkCustomer@RBS1@RBS2@RBS3)
    Call z_InsertEmptyCols("SmC_MasterDataSet_PiLevel", 1, Col_to)
    Call z_AddColNames(Array("Pild&PiStrategicCrop&TkStatus&TkCustomer&RBS1&RBS2&RBS3"),
"SmC_MasterDataSet_PiLevel", 1, Col_to)
    Col_Pild_Pi = z_GetColumnIndex("Pildentifier", 1, "SmC_MasterDataSet_PiLevel")
    Col_PiStrategicCrop_Pi = z_GetColumnIndex("PI Strategic Crop", 1, "SmC_MasterDataSet_PiLevel")
    Col_TkStatus_Pi = z_GetColumnIndex("TK Task Status", 1, "SmC_MasterDataSet_PiLevel")
    Col_TkCustomer_Pi = z_GetColumnIndex("TK Task Customer", 1, "SmC_MasterDataSet_PiLevel")
    Col_RBS1_Pi = z_GetColumnIndex("RBS Level 1", 1, "SmC_MasterDataSet_PiLevel")
    Col_RBS2_Pi = z_GetColumnIndex("RBS Level 2", 1, "SmC_MasterDataSet_PiLevel")
    Col_RBS3_Pi = z_GetColumnIndex("RBS Level 3", 1, "SmC_MasterDataSet_PiLevel")
    RowSize_Tk = z_RowSize(1, "SmC_MasterDataSet_PiLevel")
    For Row = 2 To RowSize_Tk
        Cells(Row, Col_to) = Cells(Row, Col_Pild_Pi) & "@" & Cells(Row, Col_PiStrategicCrop_Pi) & "@"
& Cells(Row, Col_TkStatus_Pi) & "@" & _
        Cells(Row, Col_TkCustomer_Pi) & "@" & Cells(Row, Col_RBS1_Pi) & "@" &
Cells(Row, Col_RBS2_Pi) & "@" & _
        Cells(Row, Col_RBS3_Pi)
    Next
    'add the concatenated key in Sheet from (position not important since sheet will be deleted
afterwards)
    Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_13", 1, Col_to)
    Call z_AddColNames(Array("Pild&PiStrategicCrop&TkStatus&TkCustomer&RBS1&RBS2&RBS3"),
"PMEC_VBA_MasterDataSet_13", 1, Col_to)
    Col_Pild = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_13")

```

```

Col_PIStrategicCrop = z_GetColumnIndex("PI Strategic Crop", 1, "PMEC_VBA_MasterDataSet_13")
col_TkStatus = z_GetColumnIndex("TK Task Status", 1, "PMEC_VBA_MasterDataSet_13")
Col_TkCustomer = z_GetColumnIndex("TK Task Customer", 1, "PMEC_VBA_MasterDataSet_13")
Col_RBS1 = z_GetColumnIndex("RBS Level 1", 1, "PMEC_VBA_MasterDataSet_13")
Col_RBS2 = z_GetColumnIndex("RBS Level 2", 1, "PMEC_VBA_MasterDataSet_13")
Col_RBS3 = z_GetColumnIndex("RBS Level 3", 1, "PMEC_VBA_MasterDataSet_13")
RowSize = z_RowSize(1, "PMEC_VBA_MasterDataSet_13")
For Row = 2 To RowSize
    Cells(Row, Col_to) = Cells(Row, Col_Pild) & "@" & Cells(Row, Col_PIStrategicCrop) & "@" &
Cells(Row, col_TkStatus) & "@" & _
        Cells(Row, Col_TkCustomer) & "@" & Cells(Row, Col_RBS1) & "@" & Cells(Row,
Col_RBS2) & "@" & _
        Cells(Row, Col_RBS3)
Next
'map attributes
Dim ColNames_from As Variant
Dim ColNames_to As Variant
Dim sSh_from As String
sSh_from = "PMEC_VBA_MasterDataSet_13"
ColNames_from = ColumnNames_ToMap
ColNames_to = ColumnNames_ToMap
'call the map function
Call z_ShMapColumns_FastVersion(sSh_from,
"PIId&PIStrategicCrop&TkStatus&TkCustomer&RBS1&RBS2&RBS3", ColNames_from, _
    "SmC_MasterDataSet_PiLevel",
"PIId&PIStrategicCrop&TkStatus&TkCustomer&RBS1&RBS2&RBS3", ColNames_to)
End If
'add timestamp and pivot light
'-----
If flag Then
    'Date Columns
    date_Arr = Array("PI Planned start", "PI Planned finish", "BC First year of sales")
    Call z_ChgDateFormat("SmC_MasterDataSet_PiLevel", date_Arr, 0)
    Dim FirstYr As Integer
    FirstYr = VBA.DateTimes.year(Now()) - 1
    Call z_AddPivotTable_LightVersion("SmC_MasterDataSet_PiLevel", "PivotTable_PiLevel",
"PIIdentifier", "Pi Title", "EAC Full Costs " & CStr(FirstYr + 1) & "_c")
    Call z_AddTimeStamp("SmC_MasterDataSet_PiLevel", StampDate, 1, 1, 25)
    Call z_AddTimeStamp("PivotTable_PiLevel", StampDate, 1, 1, 25)
    Debug.Print Now()
End If
'remove no longer needed sheets
'-----
If flag Then
    Call z_ShDelete("PMEC_VBA_MasterDataSet_13", wb_new)
End If
Call z_WorkbookSave(wb_new)
wb_new.Close
Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
    Array("Main_CheckBusinessRules", "End", "task:", "file:", CStr(Now()), ""))
End Sub

Sub m_Main_PivotFlat_PiAndTK_CustomizedDataSet_OneLinePerId(Optional wbdate As String)

```



```

Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
    Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
*****

Dim flag As Integer
Dim flag_Tk As Integer
Dim flag_Rs As Integer
'Dim wbdate As String
Dim reportversion_folder As String
Dim reportversion_file As String
*****

'reopen
flag = 1
flag_Tk = 1 'default=1
flag_Rs = 1 'default=1
If wbdate = Empty Then
    wbdate = "2012_10_16" & VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_" & VBA.DateTime.Day(Now()) & "2012_2_8"
End If
reportversion_folder = "_V1-0"
reportversion_file = "_V1-0"
folderdescription = ""
*****

'Open and activate an Excel workbook (and session)
Dim WbPath As String
Dim WbName As String
Dim Wb_GenPart4 As Workbook
Dim wb_new As Workbook
*****

'Adds the new workbook RD_MasterDataSet
*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForOneLinePerId_" & wbdate & reportversion_file & ".xlsb" 'for test purposes
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
End If
'Open existing workbook
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate & reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part4_" & wbdate & reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart4)
End If
'copy a sheet from another workbook
If flag Then
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart4, wb_new, "Begin")
    Sheets("PMEC_VBA_MasterDataSet_13").name = "PMEC_VBA_MasterDataSet_13_Pi"

```

```

    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart4, wb_new,
"Begin")
    Sheets("PMEC_VBA_MasterDataSet_13").name = "PMEC_VBA_MasterDataSet_13_TK"
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart4, wb_new,
"Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_GenPart4.Close False
End If
If flag Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", wb_new)
        Call z_ShDelete("Sheet2", wb_new)
        Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If
*****

'one line per PI
*****

'only keep needed columns from base data set (delete not needed columns)
'-----

If flag Then
    Dim ColumnNames_All As Variant
    ColumnNames_All = z_FillArray("PMEC_VBA_MasterDataSet_13_PI", , 1, 1)
    ,

    Dim Wb_from As Workbook
    Dim Rng_from As Range
    Dim ColumnNames_ToKeep As Variant
    Dim ColumnNames_ToKeep_1 As Variant
    Dim ColumnNames_ToKeep_2 As Variant
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_PI_TK_RS", 5,
Wb_from)
    ColumnNames_ToKeep_1 = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_PI_TK_RS",
Rng_from, "ReadACol", Wb_from)
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_PI_TK_RS", 6,
Wb_from)
    ColumnNames_ToKeep_2 = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_PI_TK_RS",
Rng_from, "ReadACol", Wb_from)
    ColumnNames_ToKeep = z_Arrays_PutTogether(ColumnNames_ToKeep_1,
ColumnNames_ToKeep_2)
    ,

    Call z_DeleteColumns3("PMEC_VBA_MasterDataSet_13_PI", ColumnNames_ToKeep,
ColumnNames_All)
End If
'aggregation via Pivot
'-----

If flag Then
    'Stampdate
    Dim StampDate As String
    StampDate = z_StampDate(wbdate)
    'Generate Pivot and flat table on PI level
    Debug.Print Now()

```

```

PivotName_PIs =
z_CreatePivot_PIs_CustomisedForOneLinePerId("PMEC_VBA_MasterDataSet_13_Pi", "Pivot_PIs",
"SmC_MasterDataSet_PiLevel")
'remove the grand total line
Call z_RemRow_WithAttrEntry("SmC_MasterDataSet_PiLevel", "PiIdentifier", "PiIdentifier", "Grand
Total", 0)
End If
'remove no longer needed sheets
'-----

If flag Then
Call z_ShDelete("PMEC_VBA_MasterDataSet_13_Pi", wb_new)
Call z_ShDelete("Pivot_PIs", wb_new)
End If
'mapping
'-----

If flag Then
'read attributes
Dim ColumnNames_ToMap As Variant
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_Pi_TK_RS", 7,
Wb_from)
ColumnNames_ToMap = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_Pi_TK_RS",
Rng_from, "ReadACol", Wb_from)
'insert attributes before Name_to/col_to
Dim Name_to As String
Name_to = ColumnNames_ToKeep_2(0)
Dim Col_to As Long
Col_to = z_GetColumnIndex(Name_to, 1, "SmC_MasterDataSet_PiLevel")
Dim NofCols As Integer
NofCols = UBound(ColumnNames_ToMap) - LBound(ColumnNames_ToMap) + 1
Call z_InsertEmptyCols("SmC_MasterDataSet_PiLevel", NofCols, Col_to)
Call z_AddColNames(ColumnNames_ToMap, "SmC_MasterDataSet_PiLevel", 1, Col_to)
'map attributes
Dim ColNames_from As Variant
Dim ColNames_to As Variant
ColNames_from = ColumnNames_ToMap
ColNames_to = ColumnNames_ToMap
Dim sSh_from As String
sSh_from = "PMEC_VBA_MasterDataSet_13"
'call the map function
Call z_ShMapColumns_FastVersion(sSh_from, "PiIdentifier", ColNames_from,
"SmC_MasterDataSet_PiLevel", _
"PiIdentifier", ColNames_to)
End If
'add timestamp and pivot light
'-----

If flag Then
'Date Columns
date_Arr = Array("PI Planned start", "PI Planned finish", "BC First year of sales")
Call z_ChgDateFormat("SmC_MasterDataSet_PiLevel", date_Arr, 0)

Dim FirstYr As Integer
FirstYr = VBA.DateTime.year(Now()) - 1

```

```

    Call z_AddPivotTable_LightVersion("SmC_MasterDataSet_PiLevel", "PivotTable_PiLevel",
    "PiIdentifier", "Pi Title", "EAC Full Costs " & CStr(FirstYr + 1) & "_c")
    Call z_AddTimeStamp("SmC_MasterDataSet_PiLevel", StampDate, 1, 1, 25)
    Call z_AddTimeStamp("PivotTable_PiLevel", StampDate, 1, 1, 25)
    Debug.Print Now()
End If
*****

'one line per TK
*****

'only keep needed columns from base data set (delete not needed columns)
'-----

If flag Then
    ColumnNames_All = z_FillArray("PMEC_VBA_MasterDataSet_13_TK", , 1, 1)
    ,

    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_PI_TK_RS", 11,
Wb_from)
    ColumnNames_ToKeep_1 = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_PI_TK_RS",
Rng_from, "ReadACol", Wb_from)
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_PI_TK_RS", 12,
Wb_from)
    ColumnNames_ToKeep_2 = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_PI_TK_RS",
Rng_from, "ReadACol", Wb_from)
    ColumnNames_ToKeep = z_Arrays_PutTogether(ColumnNames_ToKeep_1,
ColumnNames_ToKeep_2)
    ,

    Call z_DeleteColumns3("PMEC_VBA_MasterDataSet_13_TK", ColumnNames_ToKeep,
ColumnNames_All)
End If
'aggregation via Pivot
'-----

If flag_Tk Then
    'Stampdate
    StampDate = z_StampDate(wbdate)
    'Generate Pivot and flat table on TK level
    Debug.Print Now()
    PivotName_TKs =
z_CreatePivot_TKs_CustomisedForOneLinePerId("PMEC_VBA_MasterDataSet_13_TK", "Pivot_TKs",
"SmC_MasterDataSet_TkLevel")
    'remove the grand total line
    Call z_RemRow_WithAttrEntry("SmC_MasterDataSet_TkLevel", "TkIdentifier", "TkIdentifier",
"Grand Total", 0)
End If
'remove no longer needed sheets
'-----

If flag Then
    Call z_ShDelete("PMEC_VBA_MasterDataSet_13_TK", wb_new)
    Call z_ShDelete("Pivot_TKs", wb_new)
End If
'mapping
'-----

If flag Then
    'read attributes

```

```

Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_PI_TK_RS", 13,
Wb_from)
ColumnNames_ToMap = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_PI_TK_RS",
Rng_from, "ReadACol", Wb_from)
'insert attributes before Name_to/col_to
Name_to = ColumnNames_ToKeep_2(0)
Col_to = z_GetColumnIndex(Name_to, 1, "SmC_MasterDataSet_TkLevel")
NofCols = UBound(ColumnNames_ToMap) - LBound(ColumnNames_ToMap) + 1
Call z_InsertEmptyCols("SmC_MasterDataSet_TkLevel", NofCols, Col_to)
Call z_AddColNames(ColumnNames_ToMap, "SmC_MasterDataSet_TkLevel", 1, Col_to)
'add the concatenated key in Sheet to
Call z_InsertEmptyCols("SmC_MasterDataSet_TkLevel", 1, Col_to)
Call z_AddColNames(Array("TkIdentifier&WsIdentifier&PIdentifier"),
"SmC_MasterDataSet_TkLevel", 1, Col_to)
Col_Pild_TK = z_GetColumnIndex("PIdentifier", 1, "SmC_MasterDataSet_TkLevel")
Col_TkId_TK = z_GetColumnIndex("TkIdentifier", 1, "SmC_MasterDataSet_TkLevel")
Col_WsId_TK = z_GetColumnIndex("WsIdentifier", 1, "SmC_MasterDataSet_TkLevel")
RowSize_Tk = z_RowSize(1, "SmC_MasterDataSet_TkLevel")
For Row = 2 To RowSize_Tk
    Cells(Row, Col_to) = Cells(Row, Col_TkId_TK) & "@" & Cells(Row, Col_WsId_TK) & "@" &
Cells(Row, Col_Pild_TK)
Next
'add the concatenated key in Sheet from (position not important since sheet will be deleted
afterwards)
Call z_InsertEmptyCols("PMEC_VBA_MasterDataSet_13", 1, Col_to)
Call z_AddColNames(Array("TkIdentifier&WsIdentifier&PIdentifier"),
"PMEC_VBA_MasterDataSet_13", 1, Col_to)
Col_Pild = z_GetColumnIndex("PIdentifier", 1, "PMEC_VBA_MasterDataSet_13")
col_TkId = z_GetColumnIndex("TkIdentifier", 1, "PMEC_VBA_MasterDataSet_13")
col_WsId = z_GetColumnIndex("WsIdentifier", 1, "PMEC_VBA_MasterDataSet_13")
RowSize = z_RowSize(1, "PMEC_VBA_MasterDataSet_13")
For Row = 2 To RowSize
    Cells(Row, Col_to) = Cells(Row, col_TkId) & "@" & Cells(Row, col_WsId) & "@" & Cells(Row,
Col_Pild)
Next
'map attributes
sSh_from = "PMEC_VBA_MasterDataSet_13"
ColNames_from = ColumnNames_ToMap
ColNames_to = ColumnNames_ToMap
'call the map function
Call z_ShMapColumns_FastVersion(sSh_from, "TkIdentifier&WsIdentifier&PIdentifier",
ColNames_from, _
    "SmC_MasterDataSet_TkLevel", "TkIdentifier&WsIdentifier&PIdentifier", ColNames_to)
' 'delete TkIdentifier=(blank)
' RowSize = z_RowSize(1, "SmC_MasterDataSet_TkLevel")
' For Row = 2 To 1000
'     If Cells(Row, 1) = "(blank)" Then
'         Cells(Row, 1).EntireRow.Delete
'         Row = Row - 1
'     End If
' Next
End If
'add timestamp and pivot light

```

```

'-----
If flag Then
    'Date Columns
    date_Arr = Array("PI Planned start", "PI Planned finish", "Activity Planned Start", "Activity Expected
finish", "Activity Actual Start", "Activity Actual Finish", "BC First year of sales")
    Call z_ChgDateFormat("SmC_MasterDataSet_TkLevel", date_Arr, 0)

    FirstYr = VBA.DateTime.year(Now()) - 1
    Call z_AddPivotTable_LightVersion("SmC_MasterDataSet_TkLevel", "PivotTable_TkLevel",
"TkIdentifier", "TK Task Title", "EAC Full Costs " & CStr(FirstYr + 1) & "_c")
    Call z_AddTimeStamp("SmC_MasterDataSet_TkLevel", StampDate, 1, 1, 25)
    Call z_AddTimeStamp("PivotTable_TkLevel", StampDate, 1, 1, 25)
    Debug.Print Now()
End If
'remove no longer needed sheets
'-----
If flag Then
    Call z_ShDelete("PMEC_VBA_MasterDataSet_13", wb_new)
End If

    Call z_WorkbookSave(wb_new)
    wb_new.Close
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "End", "task:", "file:", CStr(Now()), ""))
End Sub

Sub m_Main_PivotFlat_Rs_CustomizedDataSet_OneLinePerId(Optional wbdate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
    *****

    Dim flag As Integer
    Dim flag_Tk As Integer
    Dim flag_Rs As Integer
    'Dim wbdate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    flag_Rs = 1 'default=1
    If wbdate = Empty Then
        wbdate = "2012_10_16" 'VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) "'2012_2_8"
    End If
    reportversion_folder = "_V1-0"
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String
    Dim Wb_GenPart4 As Workbook
    Dim wb_new As Workbook

```

```

*****
*****

'Adds the new workbook RD_MasterDataSet
*****
*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForOneLinePerId2_" &
wbdate & reportversion_file & ".xlsb" 'for test purposes
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
End If
'Open existing workbook
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part4_" & wbdate &
reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart4)
End If
'copy a sheet from another workbook
If flag Then
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart4, wb_new,
"Begin")
    Sheets("PMEC_VBA_MasterDataSet_13").name = "PMEC_VBA_MasterDataSet_13_RS"
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_13", Wb_GenPart4, wb_new,
"Begin")
    Wb_GenPart4.Close False
End If
If flag Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
    Call z_ShDelete("Sheet1", wb_new)
    Call z_ShDelete("Sheet2", wb_new)
    Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If
*****

'one line per RS
*****

'only keep needed columns from base data set (delete not needed columns)
'-----

If flag Then
    Dim ColumnNames_All As Variant
    ColumnNames_All = z_FillArray("PMEC_VBA_MasterDataSet_13_RS", , 1, 1)
    ,

    Dim Wb_from As Workbook
    Dim Rng_from As Range
    Dim ColumnNames_ToKeep As Variant
    Dim ColumnNames_ToKeep_1 As Variant
    Dim ColumnNames_ToKeep_2 As Variant

```

```

    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_PI_TK_RS", 14,
Wb_from)
    ColumnNames_ToKeep_1 = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_PI_TK_RS",
Rng_from, "ReadACol", Wb_from)
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_PI_TK_RS", 15,
Wb_from)
    ColumnNames_ToKeep_2 = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_PI_TK_RS",
Rng_from, "ReadACol", Wb_from)
    ColumnNames_ToKeep = z_Arrays_PutTogether(ColumnNames_ToKeep_1,
ColumnNames_ToKeep_2)
    '
    Call z_DeleteColumns3("PMEC_VBA_MasterDataSet_13_RS", ColumnNames_ToKeep,
ColumnNames_All)
End If
'aggregation via Pivot
'-----
If flag Then
    'Stampdate
    Dim StampDate As String
    StampDate = z_StampDate(wbdate)
    'Generate Pivot and flat table on TK level
    Debug.Print Now()
    PivotName_RSs =
z_CreatePivot_RSs_CustomisedForOneLinePerId("PMEC_VBA_MasterDataSet_13_RS", "Pivot_RSs",
"SmC_MasterDataSet_ResourceLevel")
    'remove the grand total line
    Call z_RemRow_WithAttrEntry("SmC_MasterDataSet_ResourceLevel", "RsIdentifier",
"RsIdentifier", "Grand Total", 0)
End If
'remove no longer needed sheets
'-----
If flag Then
    Call z_ShDelete("PMEC_VBA_MasterDataSet_13_RS", wb_new)
    Call z_ShDelete("Pivot_RSs", wb_new)
End If
'mapping
'-----
If flag Then
    'read attributes
    Dim ColumnNames_ToMap As Variant
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("PivotFlat_PI_TK_RS", 16,
Wb_from)
    ColumnNames_ToMap = z_ReadAttributeNames_FromSheet_ToArray("PivotFlat_PI_TK_RS",
Rng_from, "ReadACol", Wb_from)
    'insert attributes before Name_to/col_to
    Dim Name_to As String
    Name_to = ColumnNames_ToKeep_2(0)
    Dim Col_to As Long
    Col_to = z_GetColumnIndex(Name_to, 1, "SmC_MasterDataSet_ResourceLevel")
    Dim NofCols As Integer
    NofCols = UBound(ColumnNames_ToMap) - LBound(ColumnNames_ToMap) + 1
    Call z_InsertEmptyCols("SmC_MasterDataSet_ResourceLevel", NofCols, Col_to)
    Call z_AddColNames(ColumnNames_ToMap, "SmC_MasterDataSet_ResourceLevel", 1, Col_to)

```



```

'map attributes
Dim ColNames_from As Variant
Dim ColNames_to As Variant
ColNames_from = ColumnNames_ToMap
ColNames_to = ColumnNames_ToMap
Dim sSh_from As String
sSh_from = "PMEC_VBA_MasterDataSet_13"
'call the map function
Call z_ShMapColumns_FastVersion(sSh_from, "RsIdentifier", ColNames_from,
"SmC_MasterDataSet_ResourceLevel", _
    "RsIdentifier", ColNames_to)
End If
'add timestamp and pivot light
'-----
If flag_Rs Then
    'Date Columns
    date_Arr = Array("PI Planned start", "PI Planned finish", "Activity Planned Start", "Activity Expected
finish", "Activity Actual Start", "Activity Actual Finish", "BC First year of sales")
    Call z_ChgDateFormat("SmC_MasterDataSet_ResourceLevel", date_Arr, 0)

    Dim FirstYr As Integer
    FirstYr = VBA.DateTime.year(Now()) - 1
    Call z_AddPivotTable_LightVersion("SmC_MasterDataSet_ResourceLevel", "PivotTable_RsLevel",
"RsIdentifier", "EAC Full Costs " & CStr(FirstYr + 1) & "_c")
    Call z_AddTimeStamp("SmC_MasterDataSet_ResourceLevel", StampDate, 1, 1, 25)
    Call z_AddTimeStamp("PivotTable_RsLevel", StampDate, 1, 1, 25)
    Debug.Print Now()
End If
'remove no longer needed sheets
'-----
If flag Then
    Call z_ShDelete("PMEC_VBA_MasterDataSet_13", wb_new)
End If
    Call z_WorkbookSave(wb_new)
    wb_new.Close
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "End", "task:", "file:", CStr(Now()), ""))
End Sub

```

```

Sub StartAll_CheckAttributesEntries()
    Dim wbdate As String
    wbdate = "2012_10_16"
    Call m_CheckAttributeEntries_PILevel_MappingAndColoring(wbdate)
    Call m_CheckAttributeEntries_TKLevel_MappingAndColoring(wbdate)
    'runs through if no colored cell exists otherwise a msgbox pops up
    Call m_ShowTheErrors("Pi")
    Call m_ShowTheErrors("Tk")
End Sub

Sub StartAll_CheckAttributesEntries_ArbitraryLevel()
    Call m_CheckAttributeEntries_ArbitraryLevel_MappingAndColoring

```

[illegible]

```

Set Rng_Data = Range(Cells(2, 1), Cells(RowSize, ColSize))
On Error GoTo NoCellsVisible:
    flag_NoCellsVisible = False
    Rng_Data.SpecialCells (xlCellTypeVisible)
On Error GoTo 0
If flag_NoCellsVisible <> True Then
    answer = InputBox("next", , , 1000, 10000)
    If answer <> Empty Then
        Stop
        Sheets(Sh_Comparison).Activate
    End If
End If
Call z_AutofilterClear(Sh_Comparison)
Next
Exit Sub
NoCellsVisible:
    flag_NoCellsVisible = True
    Resume Next
End Sub

Sub m_CheckAttributeEntries_TwoIdenticalSheets()
'-----
'Use this simple version if you have two sheets with identical format, same rows and column and
'the same identifier in the same row.
'-----
Dim Sh1 As String
Dim Id_Sh1 As String
Dim Id2_Sh1 As String
Dim Col_Id_Sh1 As Long
Dim Col_Id2_Sh1 As Long
Dim RowSize_Sh1 As Long
Dim ColSize_Sh1 As Long
Dim Rng_Sh1 As Range

Dim Sh2 As String
Dim Id_Sh2 As String
Dim Id2_Sh2 As String
Dim Col_Id_Sh2 As Long
Dim Col_Id2_Sh2 As Long
Dim RowSize_Sh2 As Long
Dim ColSize_Sh2 As Long
Dim Rng_Sh2 As Range

'*****

MsgBox ("Move the sheets to check manually into a workbook and adapt the sheet names")
Stop
'Sh1 will be colored!!
'Sh1 and Sh2 will be sorted on Id and Id2
Sh1 = "Sh_Result" & "SmC_MasterDataSet_ResourceLev n"
Id_Sh1 = "UC AGI Code" & "PiIdentifier"
Id2_Sh1 = "ActivityIdentifier"
Sh2 = "Sh_Result_1" & "SmC_MasterDataSet_ResourceLev o"
Id_Sh2 = "UC AGI Code" & "PiIdentifier"

```

Id2\_Sh2 = "ActivityIdentifier"

\*\*\*\*\*

Sheets(Sh1).Activate

Col\_Id\_Sh1 = z\_GetColumnIndex(Id\_Sh1, 1, Sh1)

Col\_Id2\_Sh1 = z\_GetColumnIndex(Id2\_Sh1, 1, Sh1)

RowSize\_Sh1 = z\_RowSize(Col\_Id\_Sh1, Sh1)

ColSize\_Sh1 = z\_ColSize(1, Sh1)

Set Rng\_Sh1 = Range(Cells(1, 1), Cells(RowSize\_Sh1, ColSize\_Sh1))

Sheets(Sh2).Activate

Col\_Id\_Sh2 = z\_GetColumnIndex(Id\_Sh2, 1, Sh2)

Col\_Id2\_Sh2 = z\_GetColumnIndex(Id2\_Sh2, 1, Sh2)

RowSize\_Sh2 = z\_RowSize(Col\_Id\_Sh2, Sh2)

ColSize\_Sh2 = z\_ColSize(1, Sh2)

Set Rng\_Sh2 = Range(Cells(1, 1), Cells(RowSize\_Sh2, ColSize\_Sh2))

If RowSize\_Sh1 <> RowSize\_Sh2 Or ColSize\_Sh1 <> ColSize\_Sh2 Then

    'do not use it

    Stop

End If

'sort both sheets PI, TK

Call z\_Sort(Sh1, Col\_Id\_Sh1, Col\_Id2\_Sh1, Col\_Id\_Sh1, 1)

Call z\_Sort(Sh2, Col\_Id\_Sh2, Col\_Id2\_Sh2, Col\_Id\_Sh2, 1)

'checking and coloring

Dim flag\_CaseInsensitive As Boolean

flag\_CaseInsensitive = False

For Row = 2 To RowSize\_Sh1

    For col = 1 To ColSize\_Sh1

        'case sensitive

        If flag\_CaseInsensitive = False Then

            If Rng\_Sh1(Row, col).Value <> Rng\_Sh2(Row, col).Value Then

                If Rng\_Sh1(Row, col) = "(blank)" And Rng\_Sh2(Row, col) = "" Then

                    'ok

                Else

                    'difference

                    Rng\_Sh1(Row, col).Interior.Color = 255

                End If

        End If

        'case insensitive

        Else

            If LCase(Rng\_Sh1(Row, col).Value) <> LCase(Rng\_Sh2(Row, col).Value) Then

                If Rng\_Sh1(Row, col) = "(blank)" And Rng\_Sh2(Row, col) = "" Then

                    'ok

                Else

                    'difference

                    Rng\_Sh1(Row, col).Interior.Color = 255

                End If

        End If

    End If

Next

```

Next
End Sub
Sub m_CheckAttributeEntries_ArbitraryLevel_MappingAndColoring()
'-----
'Use this sophisticated version if you have two sheets with different format.
'-----

Dim Wb_Comparison As Workbook
Dim ShToCheck As String
Dim ShRef As String
Dim ShComparison As String
Dim Key_ShToCheck As String
Dim Key_ShRef As String
Dim AttrArr_ShToCheck As Variant
Dim AttrArr_ShRef As Variant
'*****

MsgBox ("Move the sheets to check manually into a workbook and adapt the sheet names")
Stop
'ShComparison will be colored
'ShToCheck and ShRef will be sorted and timestamp removed
Set Wb_Comparison = ActiveWorkbook
ShToCheck = "before"
MsgBox ("Choose the right identifier: Tk or Activity if you have TkAttributes")
Stop
Key_ShToCheck = "TkIdentifier"
ShRef = "after"
Key_ShRef = Key_ShToCheck
'*****

Call z_CreateArrayInput_AttributeNames(ShToCheck, "CreateArrayInput_Out", 8, Wb_Comparison,
ThisWorkbook)
MsgBox ("Enter the attributes to check: use h01_CreateArrayInput")
Stop
AttrArr_ShToCheck = Array( _
    "Task Customer")
AttrArr_ShRef = AttrArr_ShToCheck
'*****

'add a new sheet
Wb_Comparison.Activate
ShComparison = "Sh_Comparison"
Call z_ShNew(ShComparison, "Before")
'delete also the formatting, z_ShNew deletes only the content
Sheets(ShComparison).Activate
Cells.Select
Selection.Delete
'Call the attribute check and coloring functions
Dim CaseInsensitive As Boolean
CaseInsensitive = True
Call z_CheckAttributeEntries_MappingAndColoring(ShToCheck, ShRef, ShComparison, _
    Key_ShToCheck, Key_ShRef, _
    AttrArr_ShToCheck, AttrArr_ShRef, CaseInsensitive)
End Sub
Sub m_CheckAttributeEntries_PILevel_MappingAndColoring(Optional wbdate As String)
'-----
'Use this sophisticated version if you have two sheets with different format.

```

```

'-----
*****

Dim flag As Integer
'Dim wbdate As String
Dim reportversion_folder As String
Dim reportversion_file As String
*****

'reopen
flag = 1
flag_OpenExistingWbs = 1
If wbdate = Empty Then
    wbdate = "2012_10_16" 'VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) '2012_2_8"
End If
reportversion_folder = "_V1-0"
reportversion_file = "_V1-0"
folderdescription = ""
*****

'Open and activate an Excel workbook (and session)
Dim WbPath As String
Dim WbName As String
Dim wb_new As Workbook
Dim Wb_FunctionalReporting As Workbook
Dim Wb_OneLinePerPi As Workbook
*****
*****

'Adds the new workbook RD_MasterDataSet
*****
*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_CheckAttributeEntries_" & wbdate
& reportversion_file & ".xlsb" 'for test purposes
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
End If
'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForFunctionalReporting_"
& wbdate & reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_FunctionalReporting)
End If
'copy a sheet from another workbook
If flag_OpenExistingWbs Then
    Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_ResourceLevel",
Wb_FunctionalReporting, wb_new, "Begin")
    Wb_FunctionalReporting.Close False
End If
'Open existing workbook
If flag_OpenExistingWbs Then

```

```

    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForOneLinePerId_" &
wbdate & reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_OneLinePerPi)
End If
    'copy a sheet from another workbook
If flag_OpenExistingWbs Then
    Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_PiLevel", Wb_OneLinePerPi, wb_new,
"Begin")
    Wb_OneLinePerPi.Close False
End If
If flag_OpenExistingWbs Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", wb_new)
        Call z_ShDelete("Sheet2", wb_new)
        Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If
*****

Dim ShToCheck As String
Dim ShRef As String
Dim ShComparison As String
Dim Key_ShToCheck As String
Dim Key_ShRef As String
Dim AttrArr_ShToCheck As Variant
Dim AttrArr_ShRef As Variant
*****

'ShComparison will be colored
'ShToCheck and ShRef will be sorted and timestamp removed
ShToCheck = "SmC_MasterDataSet_PiLevel"
Key_ShToCheck = "PiIdentifier"
ShRef = "SmC_MasterDataSet_ResourceLevel"
Key_ShRef = "PiIdentifier"
*****

'read attribute names
Dim Wb_from As Workbook
Dim Rng_from As Range
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("Chk_AttributeEntries", 2,
Wb_from)
AttrArr_ShToCheck = z_ReadAttributeNames_FromSheet_ToArray("Chk_AttributeEntries",
Rng_from, "ReadACol", Wb_from)

' AttrArr_ShToCheck = Array( _
'     "PiIdentifier", "PiIdentifier&PiTitle", "Portfolio Level 1", "Portfolio Level 2", "Portfolio Level 3",
"PI Syngenta Portfolio", "PI Syngenta Program", _
'     "PI Title", "PI Status", "PI Label", "PI Comment", "PI Manager", "PI Sponsor", "PI Stage", "PI
Scope", "PI Type", "PI Sub Type", "PI Investment Category", "PI Customer", "PI Responsibility", _
'     "PI Geography", "PI List of Regions", "PI List of Countries", "PI list of Crops Group", "PI list of
Crops", "PI Purchase Order", "PI Lead AI", "PI List of Active Ingredients", "PI Last Gate Passed", "PI
Planned start", "PI Planned finish", "PI Product Concept", "PI Is confidential ?", _

```

```

' "BC Business Case Author", "BC Total NPV", "BC First year of sales", _
' "BC Terminal Value NPV (10% decline)", "BC Sales Peak", "BC Required" _
' )
AttrArr_ShRef = AttrArr_ShToCheck
*****

'add a new sheet
ShComparison = "Sh_PiLevel_Comparison"
Call z_ShNew(ShComparison, "Before")
'delete also the formatting, z_ShNew deletes only the content
Sheets(ShComparison).Activate
Cells.Select
Selection.Delete
'Call the attribute check and coloring functions
Dim CaseInsensitive As Boolean
CaseInsensitive = True
Call z_CheckAttributeEntries_MappingAndColoring(ShToCheck, ShRef, ShComparison, _
    Key_ShToCheck, Key_ShRef, _
    AttrArr_ShToCheck, AttrArr_ShRef, CaseInsensitive)
Call z_CheckEACTotals(ShToCheck, ShRef, ShComparison, _
    Key_ShToCheck, Key_ShRef, _
    "EAC # SD 2011_c", "EAC Ext $ Direct Costs_c")
End Sub
Sub m_CheckAttributeEntries_TKLevel_MappingAndColoring(Optional wbdate As String)
'-----
'Use this sophisticated version if you have two sheets with different format.
'-----
*****

Dim flag As Integer
'Dim wbdate As String
Dim reportversion_folder As String
Dim reportversion_file As String
*****

'reopen
flag = 1
flag_OpenExistingWbs = 1
If wbdate = Empty Then
    wbdate = "2012_10_16" 'VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) "'2012_2_8"
End If
reportversion_folder = "_V1-0"
reportversion_file = "_V1-0"
folderdescription = ""
*****

'Open and activate an Excel workbook (and session)
Dim WbPath As String
Dim WbName As String
Dim wb_new As Workbook
Dim Wb_FunctionalReporting As Workbook
Dim Wb_OneLinePerTk As Workbook
*****
*****

'Adds the new workbook RD_MasterDataSet
*****

```



```

*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_CheckAttributeEntries_" & wdate
& reportversion_file & ".xlsb" 'for test purposes
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
End If
'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForFunctionalReporting_"
& wdate & reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_FunctionalReporting)
End If
'copy a sheet from another workbook
If flag_OpenExistingWbs Then
    'check first if sheet already exists
    Dim flag_SheetExists As Boolean
    flag_SheetExists = z_CheckSheetExistence(wb_new, "SmC_MasterDataSet_ResourceLevel")
    If flag_SheetExists = False Then
        Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_ResourceLevel",
Wb_FunctionalReporting, wb_new, "Begin")
    End If
    Wb_FunctionalReporting.Close False
End If
'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForOneLinePerId_" &
wdate & reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_OneLinePerTk)
End If
'copy a sheet from another workbook
If flag_OpenExistingWbs Then
    Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_TkLevel", Wb_OneLinePerTk, wb_new,
"Begin")
    Wb_OneLinePerTk.Close False
End If
If flag_OpenExistingWbs Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
    Call z_ShDelete("Sheet1", wb_new)
    Call z_ShDelete("Sheet2", wb_new)
    Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If
*****

Dim ShToCheck As String

```

```

Dim ShRef As String
Dim ShComparison As String
Dim Key_ShToCheck As String
Dim Key_ShRef As String
Dim AttrArr_ShToCheck As Variant
Dim AttrArr_ShRef As Variant
'*****

'ShComparison will be colored
'ShToCheck and ShRef will be sorted and timestamp removed
ShToCheck = "SmC_MasterDataSet_TkLevel"
Key_ShToCheck = "TkIdentifier"
ShRef = "SmC_MasterDataSet_ResourceLevel"
Key_ShRef = "TkIdentifier"
'*****

'read attribute names
Dim Wb_from As Workbook
Dim Rng_from As Range
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("Chk_AttributeEntries", 3,
Wb_from)
AttrArr_ShToCheck = z_ReadAttributeNames_FromSheet_ToArray("Chk_AttributeEntries",
Rng_from, "ReadACol", Wb_from)

' AttrArr_ShToCheck = Array( _
'     "PIdentifier", "WsIdentifier", "TkIdentifier", "ActivityIdentifier", "PIdentifier&PiTitle",
"ActivityIdentifier&TaskTitle", "Portfolio Level 1", "Portfolio Level 2", "Portfolio Level 3", "PI Syngenta
Portfolio", "PI Syngenta Program", _
'     "PI Title", "PI Status", "PI Label", "PI Comment", "PI Manager", "PI Sponsor", "PI Stage", "PI
Scope", "PI Type", "PI Sub Type", "PI Investment Category", "PI Customer", "PI Responsibility", _
'     "PI Geography", "PI List of Regions", "PI List of Countries", "PI list of Crops Group", "PI list of
Crops", "PI Purchase Order", "PI Lead AI", "PI List of Active Ingredients", "PI Last Gate Passed", "PI
Planned start", "PI Planned finish", "PI Product Concept", "PI Is confidential ?", "TK Task Title", _
'     "TK Task Contact", "TK Task Location", "TK Task Status", "Activity Description", "Activity
Comment", "Activity Type", "Activity Duration", "Activity Planned Start", "Activity Expected finish",
"Activity Actual Start", "Activity Actual Finish", "BC Business Case Author", "BC Total NPV", "BC First
year of sales", _
'     "BC Terminal Value NPV (10% decline)", "BC Sales Peak", "BC Required" _
' )
AttrArr_ShRef = AttrArr_ShToCheck
'*****

'add a new sheet
ShComparison = "Sh_TkLevel_Comparison"
Call z_ShNew(ShComparison, "Before")
'delete also the formatting, z_ShNew deletes only the content
Sheets(ShComparison).Activate
Cells.Select
Selection.Delete
'Call the attribute check and coloring functions
Dim CaseInsensitive As Boolean
CaseInsensitive = True
Call z_CheckAttributeEntries_MappingAndColoring(ShToCheck, ShRef, ShComparison, _
Key_ShToCheck, Key_ShRef, _
AttrArr_ShToCheck, AttrArr_ShRef, CaseInsensitive)
'remove all lines with TkIdentifier = (blank)

```

```

Dim RowSize As Long
Sheets(ShComparison).Activate
RowSize = z_RowSize(1, ShComparison)
For iter = 1 To RowSize
    If Cells(iter, 1).Value = "(blank)" Then
        Cells(iter, 1).Select
        Selection.EntireRow.Delete
        iter = iter - 1
    End If
Next
Call z_CheckEACTotals(ShToCheck, ShRef, ShComparison, _
    Key_ShToCheck, Key_ShRef, _
    "EAC # SD 2011_c", "EAC Ext $ Direct Costs_c")
End Sub
Function z_CheckEACTotals( _
    Sh_ToCheck As String, Sh_Reference As String, Sh_Comparison As String, _
    KeyName_ToCheck As String, KeyName_Reference As String, _
    firstEACName As String, lastEACName As String)
'get column indices of the keys in Sh_ToCheck and Sh_Reference
Dim Col_Key_Comparison As Long
Dim Col_KeyNameToCheck As Long
Dim Col_KeyNameReference As Long
Col_Key_Comparison = 1
Col_KeyNameToCheck = z_GetColumnIndex(KeyName_ToCheck, 1, Sh_ToCheck)
Col_KeyNameReference = z_GetColumnIndex(KeyName_Reference, 1, Sh_Reference)
'get column indices of firstEACName and lastEACName in Sh_ToCheck
Dim Col_firstEACName_ShToCheck As Long
Dim Col_lastEACName_ShToCheck As Long
Col_firstEACName_ShToCheck = z_GetColumnIndex(firstEACName, 1, Sh_ToCheck)
Col_lastEACName_ShToCheck = z_GetColumnIndex(lastEACName, 1, Sh_ToCheck)
'get column indices of firstEACName and lastEACName in Sh_Reference
Dim Col_firstEACName_ShRef As Long
Dim Col_lastEACName_ShRef As Long
Col_firstEACName_ShRef = z_GetColumnIndex(firstEACName, 1, Sh_Reference)
Col_lastEACName_ShRef = z_GetColumnIndex(lastEACName, 1, Sh_Reference)
'get lastwritten column in Sh_Comparison
Dim Col_To_ShComparison As Long
Col_To_ShComparison = z_ColSize(1, Sh_Comparison) + 1
'copy/paste the column name from Sh_ToCheck into Sh_Comparison(1. row)
Dim Range_from As Range
Sheets(Sh_ToCheck).Activate
Set Range_from = Sheets(Sh_ToCheck).Range(Cells(1, Col_firstEACName_ShToCheck), _
    Cells(1, Col_lastEACName_ShToCheck))
'version without activating the sheet
'Set Range_From = Nothing
'Set Range_From = Sheets(Sh_ToCheck).Range(Sheets(Sh_ToCheck).Cells(1,
Col_firstEACName_ShToCheck), _
    Sheets(Sh_ToCheck).Cells(1, Col_lastEACName_ShToCheck))
Call z_CopyRange("All", Sh_ToCheck, Range_from, Sh_Comparison,
Sheets(Sh_Comparison).Cells(1, Col_To_ShComparison))
'loop through the columns in Sh_ToCheck, build the sum and write it into Sh_Comparison (2. row)
Dim RowSize_ShToCheck As Long
RowSize_ShToCheck = z_RowSize(Col_KeyNameToCheck, Sh_ToCheck)

```

```

Dim Col_to As Long
Col_to = Col_To_ShComparison
For Col_from = Col_firstEACName_ShToCheck To Col_lastEACName_ShToCheck
    Sheets(Sh_ToCheck).Activate
    TotalSum = WorksheetFunction.Sum(Range(Cells(2, Col_from), Cells(RowSize_ShToCheck,
Col_from)))
    Sheets(Sh_Comparison).Cells(2, Col_to).Value = TotalSum
    Col_to = Col_to + 1
Next
'loop through the columns in Sh_Reference, build the sum and write it into Sh_Comparison (3.
row)
Dim RowSize_Reference As Long
RowSize_Reference = z_RowSize(Col_KeyNameReference, Sh_Reference)
Col_to = Col_To_ShComparison
For Col_from = Col_firstEACName_ShRef To Col_lastEACName_ShRef
    Sheets(Sh_Reference).Activate
    TotalSum = WorksheetFunction.Sum(Range(Cells(2, Col_from), Cells(RowSize_Reference,
Col_from)))
    Sheets(Sh_Comparison).Cells(3, Col_to).Value = TotalSum
    Col_to = Col_to + 1
Next
'in Sh_Comparison calculate the difference in 4.row between 2.row and 3.row
Col_to = Col_To_ShComparison
For Col_to = Col_To_ShComparison To Col_To_ShComparison + Col_lastEACName_ShRef -
Col_firstEACName_ShRef
    Sheets(Sh_Comparison).Activate
    Sheets(Sh_Comparison).Cells(4, Col_to).Value = Sheets(Sh_Comparison).Cells(3, Col_to).Value - _
        Sheets(Sh_Comparison).Cells(2, Col_to).Value
Next
End Function

```

```

Function z_CheckAttributeEntries_MappingAndColoring( _
    Sh_ToCheck As String, Sh_Reference As String, Sh_Comparison As String, _
    KeyName_ToCheck As String, KeyName_Reference As String, _
    AttributeArray_ToCheck As Variant, AttributeArray_Reference As Variant, _
    CaseInsensitive As Boolean)
'Copy Key_ToCheck into Sh_Comparison
Sheets(Sh_ToCheck).Activate
Dim Col_Key_Comparison As Long
Dim Col_KeyNameToCheck As Long
Col_Key_Comparison = 1
Col_KeyNameToCheck = z_GetColumnIndex(KeyName_ToCheck, 1, Sh_ToCheck)
Call z_CopyColumn(Sh_ToCheck, Col_KeyNameToCheck, Sh_Comparison, Col_Key_Comparison)
'Write Column names
Sheets(Sh_Comparison).Activate
Dim Attribute_iter As Integer
For Attribute_iter = LBound(AttributeArray_ToCheck) To UBound(AttributeArray_ToCheck)
    'Get the next attribute
    Dim AttributeToCheck_i As String
    Dim AttributeReference_i As String
    AttributeToCheck_i = AttributeArray_ToCheck(Attribute_iter)
    AttributeReference_i = AttributeArray_Reference(Attribute_iter)
    Dim Col_to As Long

```

```

        Col_to = z_ColSize(1, Sh_Comparison) + 1
        Call z_AddColNames(Array(AttributeToCheck_i & "_ToCheck", AttributeReference_i & "_Ref"),
Sh_Comparison, 1, Col_to)
    Next
'Mapping
'*****

'sort does not work with merged timestamp therefore remove it
Sheets(Sh_ToCheck).Activate
If Range("A1") Like "Data as of*" Or Range("A2") Like "Data as of*" Then
    Range("A1").Select
    Selection.EntireColumn.Delete
End If
'sort Sh_from on the column Key_from
Col_KeyNameToCheck = z_GetColumnIndex(KeyName_ToCheck, 1, Sh_ToCheck)
Call z_Sort(Sh_ToCheck, Col_KeyNameToCheck, 0, Col_KeyNameToCheck, 1)
'Fast Map AttributeArray_Reference_i for i = 1 to n into ShComparison
Dim AttributeArray_Reference_Comparison As Variant
AttributeArray_Reference_Comparison = AttributeArray_Reference
For Attribute_iter = LBound(AttributeArray_Reference) To UBound(AttributeArray_Reference)
    AttributeArray_Reference_Comparison(Attribute_iter) =
AttributeArray_Reference_Comparison(Attribute_iter) & "_Ref"
Next
Dim KeyName_Reference_Comparison As String
KeyName_Reference_Comparison = KeyName_ToCheck & "_"
Sheets(Sh_Comparison).Cells(1, Col_Key_Comparison).Value = KeyName_Reference_Comparison
Call z_ShMapColumns_FastVersion(Sh_Reference, KeyName_Reference,
AttributeArray_Reference, _
    Sh_Comparison, KeyName_Reference_Comparison, AttributeArray_Reference_Comparison, , , 1,
1)

'copy AttributeArray_ToCheck_i for i = 1 to n, and start the check of each cell
For Attribute_iter = LBound(AttributeArray_ToCheck) To UBound(AttributeArray_ToCheck)
    'Get the next attribute
    Dim Col_AttributeToCheck_i As Long
    Dim Col_AttributeReference_i As Long
    AttributeToCheck_i = AttributeArray_ToCheck(Attribute_iter)
    AttributeReference_i = AttributeArray_Reference(Attribute_iter)
    Col_AttributeToCheck_i = z_GetColumnIndex(AttributeToCheck_i, 1, Sh_ToCheck)
    Col_AttributeReference_i = z_GetColumnIndex(AttributeReference_i, 1, Sh_Reference)
    'Copy AttributeArray_ToCheck_i into ShComparison
    Dim Col_AttributeToCheck_i_Comparison As Long
    Dim Col_AttributeReference_i_Comparison As Long
    Col_AttributeToCheck_i_Comparison = z_GetColumnIndex(AttributeToCheck_i & "_ToCheck", 1,
Sh_Comparison)
    Col_AttributeReference_i_Comparison = z_GetColumnIndex(AttributeToCheck_i & "_Ref", 1,
Sh_Comparison)
    Call z_CopyColumn(Sh_ToCheck, Col_AttributeToCheck_i, Sh_Comparison,
Col_AttributeToCheck_i_Comparison)
    'Apply ConditionalFormatting on Attribute_ToCheck_i
    Call z_CheckAttributeEntries_Coloring(Sh_Comparison, Col_Key_Comparison, _
        Col_AttributeToCheck_i_Comparison, "NotEqual", Col_AttributeReference_i_Comparison,
-
        CaseInsensitive)

```

Next

End Function

```
Function z_CheckAttributeEntries_Coloring(Sh_Comparison As String, ColKey As Long, _
    Col_AttributeToCheck As Long, Condition As String, Col_AttributeReference As Long, _
    CaseInsensitive As Boolean)
    'All ranges and checks are in the Sh_Comparison
    Dim Rng_ToCheck As Range
    Dim Rng_Ref As Range
    Sheets(Sh_Comparison).Activate
    RowSize = z_RowSize(ColKey, Sh_Comparison)
    Set Rng_ToCheck = Range(Cells(1, Col_AttributeToCheck), Cells(RowSize, Col_AttributeToCheck))
    Rng_ToCheck.Select
    Set Rng_Ref = Range(Cells(1, Col_AttributeReference), Cells(RowSize, Col_AttributeReference))
    Rng_Ref.Select
    For Row = 2 To RowSize
        Dim Value_ToCheck As String
        Dim Value_Ref As String
        Dim LCaseValue_ToCheck As String
        Dim LCaseValue_Ref As String
        'case sensitive
        If CaseInsensitive = False Then
            Value_ToCheck = Rng_ToCheck(Row, 1).Value
            Value_Ref = Rng_Ref(Row, 1).Value
            If Value_ToCheck <> Value_Ref Then
                If Rng_ToCheck(Row, 1).Value = (blank) And Rng_Ref(Row, 1).Value = "" Then
                    'ok
                Else
                    'difference
                    Rng_ToCheck(Row, 1).Interior.Color = 255
                End If
            End If
        'case insensitive
        Else
            LCaseValue_ToCheck = LCase(Rng_ToCheck(Row, 1).Value)
            LCaseValue_Ref = LCase(Rng_Ref(Row, 1).Value)
            If LCaseValue_ToCheck <> LCaseValue_Ref Then
                If Rng_ToCheck(Row, 1).Value = (blank) And Rng_Ref(Row, 1).Value = "" Then
                    'ok
                Else
                    'difference
                    Rng_ToCheck(Row, 1).Interior.Color = 255
                End If
            End If
        End If
    Next
End Function

Function z_CheckAttributeEntries_Coloring2(Sh_Comparison As String, ColKey As Long, _
    Col_AttributeToCheck As Long, Condition As String, Col_AttributeReference As Long, _
    CaseInsensitive As Boolean)
    'All ranges and checks are in the Sh_Comparison
    Dim Rng_ToCheck As Range
    Dim Rng_Ref As Range
```

```

Sheets(Sh_Comparison).Activate
RowSize = z_RowSize(ColKey, Sh_Comparison)
Set Rng_ToCheck = Range(Cells(1, Col_AttributeToCheck), Cells(RowSize, Col_AttributeToCheck))
Rng_ToCheck.Select
Set Rng_Ref = Range(Cells(1, Col_AttributeReference), Cells(RowSize, Col_AttributeReference))
Rng_Ref.Select
For Row = 2 To RowSize
    Dim Value_ToCheck As String
    Dim Value_Ref As String
    Dim LCaseValue_ToCheck As String
    Dim LCaseValue_Ref As String
    'case sensitive
    If CaseInsensitive = False Then
        Value_ToCheck = Rng_ToCheck(Row, 1).Value
        Value_Ref = Rng_Ref(Row, 1).Value
        If Value_ToCheck <> Value_Ref Then
            If Rng_ToCheck(Row, 1).Value = (blank) And Rng_Ref(Row, 1).Value = "" Then
                'ok
            Else
                'difference
                Rng_ToCheck(Row, 1).Interior.Color = 255
            End If
        End If
    'case insensitive
    Else
        LCaseValue_ToCheck = LCase(Rng_ToCheck(Row, 1).Value)
        LCaseValue_Ref = LCase(Rng_Ref(Row, 1).Value)

        If Application.WorksheetFunction.IsText(Rng_ToCheck(Row, 1).Value) Or Rng_ToCheck(Row,
1).Value = Empty Then
            If LCaseValue_ToCheck <> LCaseValue_Ref Then
                If Rng_ToCheck(Row, 1).Value = (blank) And Rng_Ref(Row, 1).Value = "" Then
                    'ok
                Else
                    'real difference
                    Rng_ToCheck(Row, 1).Interior.Color = 255
                End If
            End If
        ElseIf Application.WorksheetFunction.IsNumber(Rng_ToCheck(Row, 1).Value) = True Then
            If Rng_Ref(Row, 1).Value <> Empty And Rng_Ref(Row, 1).Value <> "(blank)" Then
                If LCaseValue_ToCheck > LCaseValue_Ref + 1 Or LCaseValue_ToCheck < LCaseValue_Ref -
1 Then
                    If Rng_ToCheck(Row, 1).Value = (blank) And Rng_Ref(Row, 1).Value = "" Then
                        'ok
                    Else
                        'real difference
                        Rng_ToCheck(Row, 1).Interior.Color = 255
                    End If
                End If
            End If
        Else
            'real difference
            Rng_ToCheck(Row, 1).Interior.Color = 255
        End If
    End If
End For

```

```

        Else
            'maybe a date in date format
        End If

    End If
Next
End Function

```

```

Sub m_CheckAttributeEntries_ConditionalFormatting()
    '-----
    'Not used because autofilter does not work properly on cells colored with conditional formatting
    '-----

    Dim AttributeArray_ToCheck As Variant
    Dim AttributeArray_Reference As Variant
    AttributeArray_ToCheck = Array( _
        "PIdentifier", "WsIdentifier", "TkIdentifier", "ActivityIdentifier")
    AttributeArray_Reference = AttributeArray_ToCheck
    Call z_CheckAttributeEntries_ConditionalFormatting("Sh_ToCheck", "Sh_Reference",
"Sh_Comparison", _
        "TkIdentifier", "TkIdentifier", _
        AttributeArray_ToCheck, AttributeArray_Reference, "Map")
End Sub

```

```

Function z_CheckAttributeEntries_ConditionalFormatting( _
    Sh_ToCheck As String, Sh_Reference As String, Sh_Comparison As String, _
    KeyName_ToCheck As String, KeyName_Reference As String, _
    AttributeArray_ToCheck As Variant, AttributeArray_Reference As Variant, CopyOrMap As
String)
    '-----
    'Not used because autofilter does not work properly on cells colored with conditional formatting
    '-----

    'Copy Key_ToCheck into Sh_Comparison
    Sheets(Sh_ToCheck).Activate
    Dim Col_Key_Comparison As Long
    Dim Col_KeyNameToCheck As Long
    Col_Key_Comparison = 1
    Col_KeyNameToCheck = z_GetColumnIndex(KeyName_ToCheck, 1, Sh_ToCheck)
    Call z_CopyColumn(Sh_ToCheck, Col_KeyNameToCheck, Sh_Comparison, Col_Key_Comparison)
    'Delete the conditional formatting
    Sheets(Sh_Comparison).Activate
    Cells.Select
    Selection.FormatConditions.Delete
    'Write Column names
    Sheets(Sh_Comparison).Activate
    Dim Attribute_iter As Integer
    For Attribute_iter = LBound(AttributeArray_ToCheck) To UBound(AttributeArray_ToCheck)
        'Get the next attribute
        Dim AttributeToCheck_i As String
        Dim AttributeReference_i As String
        AttributeToCheck_i = AttributeArray_ToCheck(Attribute_iter)
        AttributeReference_i = AttributeArray_Reference(Attribute_iter)
        Dim Col_to As Long
    
```



```

        Col_to = z_ColSize(1, Sh_Comparison) + 1
        Call z_AddColNames(Array(AttributeToCheck_i & "_ToCheck", AttributeReference_i & "_Ref"),
Sh_Comparison, 1, Col_to)
    Next

'Map
If CopyOrMap = "Map" Then
    'sort does not work with merged timestamp
    Sheets(Sh_ToCheck).Activate
    If Range("A1") Like "Data as of*" Or Range("A2") Like "Data as of*" Then
        Range("A1").Select
        Selection.EntireColumn.Delete
    End If
    'sort Sh_from on the column Key_from
    Col_KeyNameToCheck = z_GetColumnIndex(KeyName_ToCheck, 1, Sh_ToCheck)
    Call z_Sort(Sh_ToCheck, Col_KeyNameToCheck, 0, Col_KeyNameToCheck, 1)
    'Fast Map AttributeArray_Reference_i into ShComparison
    Dim AttributeArray_Reference_Comparison As Variant
    AttributeArray_Reference_Comparison = AttributeArray_Reference
    For Attribute_iter = LBound(AttributeArray_Reference) To UBound(AttributeArray_Reference)
        AttributeArray_Reference_Comparison(Attribute_iter) =
AttributeArray_Reference_Comparison(Attribute_iter) & "_Ref"
    Next
    Dim KeyName_Reference_Comparison As String
    KeyName_Reference_Comparison = KeyName_ToCheck
    Call z_ShMapColumns_FastVersion(Sh_Reference, KeyName_Reference,
AttributeArray_Reference, _
        Sh_Comparison, KeyName_Reference_Comparison, AttributeArray_Reference_Comparison, , ,
1, 1)
    ElseIf CopyOrMap = "Copy" Then
    Else
        Stop
    End If
    For Attribute_iter = LBound(AttributeArray_ToCheck) To UBound(AttributeArray_ToCheck)
        'Get the next attribute
        Dim Col_AttributeToCheck_i As Long
        Dim Col_AttributeReference_i As Long
        AttributeToCheck_i = AttributeArray_ToCheck(Attribute_iter)
        AttributeReference_i = AttributeArray_Reference(Attribute_iter)
        Col_AttributeToCheck_i = z_GetColumnIndex(AttributeToCheck_i, 1, Sh_ToCheck)
        Col_AttributeReference_i = z_GetColumnIndex(AttributeReference_i, 1, Sh_Reference)

        If CopyOrMap = "Map" Then
            'Copy AttributeArray_ToCheck_i into ShComparison
            Dim Col_AttributeToCheck_i_Comparison As Long
            Dim Col_AttributeReference_i_Comparison As Long
            Col_AttributeToCheck_i_Comparison = z_GetColumnIndex(AttributeToCheck_i & "_ToCheck",
1, Sh_Comparison)
            Col_AttributeReference_i_Comparison = z_GetColumnIndex(AttributeToCheck_i & "_Ref", 1,
Sh_Comparison)
            Call z_CopyColumn(Sh_ToCheck, Col_AttributeToCheck_i, Sh_Comparison,
Col_AttributeToCheck_i_Comparison)
            ElseIf CopyOrMap = "Copy" Then

```

```

        Col_AttributeReference_i_Comparison = z_ColSize(1, Sh_Comparison) + 1
        Call z_CopyColumn(Sh_Reference, Col_AttributeReference_i, Sh_Comparison,
Col_AttributeReference_i_Comparison)
    Else
        Stop
    End If
    'Apply ConditionalFormatting on Attribute_ToCheck_i
    Call z_ConditionalFormatting(Sh_Comparison, Col_Key_Comparison, _
        Col_AttributeToCheck_i_Comparison, "NotEqual", Col_AttributeReference_i_Comparison)
Next
End Function
Sub m_ConditionalFormatting()
    '-----
    'Not used because autofilter does not work properly on cells colored with conditional formatting
    '-----
    'Delete the conditional formatting
    Sheets("Sh_Comparison").Activate
    Cells.Select
    Selection.FormatConditions.Delete
    Call z_ConditionalFormatting("Sh_Comparison", 2, _
        3, "NotEqual", 14)
End Sub
Function z_ConditionalFormatting(Sh As String, ColKey As Long, _
    Col_AttributeToCheck As Long, Condition As String, Col_AttributeReference As Long)
    '-----
    'Not used because autofilter does not work properly on cells colored with conditional formatting
    '-----
    Sheets(Sh).Select
    Dim Address As String
    Dim Address1 As Variant
    Dim col As String
    Dim Row As String
    If Condition = "NotEqual" Then
        'Determine the Address for the formula
        Address = Cells(2, Col_AttributeReference).Address
        Address1 = Split(Address, "$")
        col = Address1(1)
        Row = Address1(2)
        'select first cell and set the conditional formatting
        Cells(2, Col_AttributeToCheck).Select
        Selection.FormatConditions.Add Type:=xlCellValue, Operator:=xlNotEqual, _
            Formula1:="=" & col & Row
        Selection.FormatConditions(Selection.FormatConditions.count).SetFirstPriority
        With Selection.FormatConditions(1).Interior
            .PatternColorIndex = xlAutomatic
            .Color = 255
            .TintAndShade = 0
        End With
        Selection.FormatConditions(1).StopIfTrue = False
        'copy /paste the formating
        Cells(2, Col_AttributeToCheck).Select
        Selection.Copy
        RowSize = z_RowSize(ColKey, Sh)
    
```

```

Range(Cells(3, Col_AttributeToCheck), Cells(RowSize, Col_AttributeToCheck)).Select
Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
'
'when a condition is true add an interior color
'
Dim Rng As Range
Set Rng = Range(Cells(2, Col_AttributeToCheck), Cells(RowSize, Col_AttributeToCheck))
Dim Cell As Range
For Each Cell In Rng
    Cell.Select
    a = z_ColorOfCF(Cell, False)
    If z_ColorOfCF(Cell, False) = 255 Then
        'Cell.FormatConditions.Delete
        Cell.Interior.Color = 235
    End If
    'b = Cell.Interior.Color
Next
Stop
Range(Cells(2, Col_AttributeToCheck), Cells(RowSize,
Col_AttributeToCheck)).FormatConditions.Delete
Elseif Condition = "NotBetween" Then
'Determine the Address for the formula
Address = Cells(2, Col_AttributeReference).Address
Address1 = Split(Address, "$")
col = Address1(1)
Row = Address1(2)
'select first cell and set the conditional formatting
Cells(2, Col_AttributeToCheck).Select
Selection.FormatConditions.Add Type:=xlCellValue, Operator:=xlNotBetween, _
Formula1:="<=E2-10", Formula2:="<=E2+10"
Selection.FormatConditions(Selection.FormatConditions.count).SetFirstPriority
With Selection.FormatConditions(1).Interior
    .PatternColorIndex = xlAutomatic
    .Color = 255
    .TintAndShade = 0
End With
Selection.FormatConditions(1).StopIfTrue = False
'copy /paste the formating
Cells(2, Col_AttributeToCheck).Select
Selection.Copy
RowSize = z_RowSize(ColKey, Sh)
Range(Cells(3, Col_AttributeToCheck), Cells(RowSize, Col_AttributeToCheck)).Select
Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
Else
    Stop
End If
End Function

```

'For the new filter rules from 03 Sept 2012

\*\*\*\*\*

Private Sub StartAll\_FilterOut()

```

Dim wbdate As String
wbdate = "2012_10_16"
Call m_Main_ApplyFilteringRules_RemoveFutureCosts_ForPortfolioReporting(wbdate)
End Sub

Sub m_Main_ApplyFilteringRules_RemoveFutureCosts_ForPortfolioReporting(Optional wbdate As
String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
    *****

    Dim flag As Integer
    'Dim wbdate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    If wbdate = Empty Then
        wbdate = "2012_10_16" & VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) & "2012_2_8"
    End If
    reportversion_folder = "_V1-0"
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String
    Dim Wb_Filtered As Workbook
    Dim Wb_PivotFlat As Workbook
    Dim Wb_AdvancedFilters As Workbook
    Dim Wb_Baseline As Workbook
    *****
    *****

    'Adds the new workbook RD_MasterDataSet
    *****
    *****

    'Add a new workbook and move the sheet SmC_MasterDataSet_PiLevel to it
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_ForPortfolioReportingFilterOut_" &
wbdate & reportversion_file & ".xlsb"
        Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, Wb_Filtered)
    End If
    'Open existing workbook
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForPortfolioReporting_"
& wbdate & reportversion_file & ".xlsb"
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_PivotFlat)
    End If

```

```

'copy a sheet from another workbook
If flag Then
    Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_PiLevel", Wb_PivotFlat, Wb_Filtered,
"Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_PivotFlat.Close False
End If
'Open existing workbook
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
    WbName = "AdvancedFilters_ForPIReporting.xlsm"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_AdvancedFilters)
End If
'copy a sheet from another workbook
If flag Then
    Call z_CopySheetFromWb1ToWb2("AdvancedFilters", Wb_AdvancedFilters, Wb_Filtered, "Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_AdvancedFilters.Close False
End If
If flag Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", Wb_Filtered)
        Call z_ShDelete("Sheet2", Wb_Filtered)
        Call z_ShDelete("Sheet3", Wb_Filtered)
    On Error GoTo 0
    Call z_WorkbookSave(Wb_Filtered)
End If
'ApplyFilteringRules
*****

'Add sheets
Call z_ShNew("Rem GlobSup&MS&CGM", "Begin")
Call z_ShNew("CPD NEW AI_c", "Begin")
Call z_ShNew("CPD NOT NEW AI_c", "Begin")
Call z_ShNew("CPR CTD_c", "Begin")
Call z_ShNew("CPR NOT CTD_c", "Begin")
Call z_ShNew("CP highest lev_c", "Begin")
Call z_ShNew("Seeds & BusinessDev_c", "Begin")
Call z_ShNew("SmC_MasterDataSet_PiLevel_temp", "Begin")

Dim Sh_Source As String
Sh_Source = "SmC_MasterDataSet_PiLevel"

'Apply Filters
Dim AdvancedFilter_Rng As String

'Rem GlobSup & MS & CGM
AdvancedFilter_Rng = "E3:L4"
Call z_ApplyAdvancedFilter_Copy(Sh_Source, "Rem GlobSup&MS&CGM", "AdvancedFilters",
AdvancedFilter_Rng)

'CPD NEW AI

```

```

AdvancedFilter_Rng = "E6:G10"
Call z_ApplyAdvancedFilter_Copy("Rem GlobSup&MS&CGM", "CPD NEW AI_c", "AdvancedFilters",
AdvancedFilter_Rng)

```

```

'CPD NOT NEW AI
AdvancedFilter_Rng = "E12:G16"
Call z_ApplyAdvancedFilter_Copy("Rem GlobSup&MS&CGM", "CPD NOT NEW AI_c",
"AdvancedFilters", AdvancedFilter_Rng)

```

```

'CPR CTD
AdvancedFilter_Rng = "E18:G22"
Call z_ApplyAdvancedFilter_Copy("Rem GlobSup&MS&CGM", "CPR CTD_c", "AdvancedFilters",
AdvancedFilter_Rng)

```

```

'CPR NOT CTD
AdvancedFilter_Rng = "E24:G28"
Call z_ApplyAdvancedFilter_Copy("Rem GlobSup&MS&CGM", "CPR NOT CTD_c",
"AdvancedFilters", AdvancedFilter_Rng)

```

```

'CP highest level
AdvancedFilter_Rng = "E30:H31"
Call z_ApplyAdvancedFilter_Copy("Rem GlobSup&MS&CGM", "CP highest lev_c",
"AdvancedFilters", AdvancedFilter_Rng)

```

```

'Seeds & BusinessDev
AdvancedFilter_Rng = "E33:F38"
Call z_ApplyAdvancedFilter_Copy("Rem GlobSup&MS&CGM", "Seeds & BusinessDev_c",
"AdvancedFilters", AdvancedFilter_Rng)

```

```

'DataSet_PiLevel
Sheets("SmC_MasterDataSet_PiLevel_temp").Select
For Each Sh_from In Wb_Filtered.Sheets
    'write the data from the sheets into the SmC_MasterDataSet_PiLevel_temp
    If Sh_from.name <> "SmC_MasterDataSet_PiLevel_temp" And Sh_from.name <>
"AdvancedFilters" _
        And Sh_from.name <> "SmC_MasterDataSet_PiLevel" And Sh_from.name <> "Rem
GlobSup&MS&CGM" Then
        If Right(Sh_from.name, 2) = "_c" Then
            Call z_CopySh1ToSh2_GivenColLastRow(Sh_from.name, "PiIdentifier",
"SmC_MasterDataSet_PiLevel_temp")
        End If
    End If
Next Sh_from

```

```

'remove first empty row
If Cells(1, 1).Value = Empty Then
    Cells(1, 1).EntireRow.Delete
End If

```

```

'remove additional attribute name lines
Dim RowSize As Long
RowSize = z_RowSize(1, "SmC_MasterDataSet_PiLevel_temp")
For Each Cell In Range(Cells(2, 1), Cells(RowSize, 1))

```

```

If Cell.Value = "PiIdentifier" Then
    Cell.EntireRow.Delete
End If
Next Cell

```

```

'Fill the blanks in the PL2 and PL3 columns
Call z_Level_X_portfolio("SmC_MasterDataSet_PiLevel_temp")

```

```

'flat value copy
Call z_ShNewFlatValueCopy("SmC_MasterDataSet_PiLevel_temp",
"SmC_MasterDataSet_PiLevel_fltr", "Begin")

```

'4.1 and 4.2 remove future costs for all Portfolios PIStatus=Terminated and Completed (finishes this year, reported only this year)

```

'*****
'*****

```

```

'read attribute names
Dim Wb_from As Workbook
Dim Rng As Range
Set Rng = z_RangeOfWb_AttributeNamesFromSheetToArray("PortfRep_RemCosts", 2, Wb_from)
Dim AttrArr As Variant
AttrArr = z_ReadAttributeNames_FromSheet_ToArray("PortfRep_RemCosts", Rng, "ReadACol",
Wb_from)
Call z_RemoveFutureCosts("SmC_MasterDataSet_PiLevel_fltr", AttrArr)

```

'5.1 remove all but future costs for CPD PISubType=Planned PISubType<>AInew (new projects, reported only next year)

```

'*****
'*****

```

```

Set Rng = z_RangeOfWb_AttributeNamesFromSheetToArray("PortfRep_RemCosts", 3, Wb_from)
AttrArr = z_ReadAttributeNames_FromSheet_ToArray("PortfRep_RemCosts", Rng, "ReadACol",
Wb_from)
Call z_RemovePastAndCurrentCostsForCPDNew("SmC_MasterDataSet_PiLevel_fltr", AttrArr)

```

```

'Add a pivot
'Dim FirstYr As Integer
FirstYr = VBA.DateTimes.year(Now()) - 1
Call z_AddPivotTable_LightVersion("SmC_MasterDataSet_PiLevel_fltr", "PivotTable_PiLevel_fltr",
"PiIdentifier", "Pi Title", "EAC Full Costs " & CStr(FirstYr + 1) & "_c")

```

```

'add timestamps
Dim StampDate As String
StampDate = z_StampDate(wbdate)
Call z_AddTimeStamp("SmC_MasterDataSet_PiLevel_fltr", StampDate, 1, 1, 25)
Call z_AddTimeStamp("PivotTable_PiLevel_fltr", StampDate, 1, 1, 25)

```

```

'close
Call z_WorkbookSave(Wb_Filtered)

```

Wb\_Filtered.Close True

End Sub

'with the additional comparison

\*\*\*\*\*

Private Sub StartAll\_CurrentDatasetAndBaseline()

Dim wbdate As String

wbdate = "2012\_10\_16"

Call m\_Main\_CurrentDatasetAndBaseline(wbdate)

End Sub

Sub m\_Main\_CurrentDatasetAndBaseline(Optional wbdate As String)

Call z\_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", \_

Array("Main\_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))

\*\*\*\*\*

Dim flag As Integer

'Dim wbdate As String

Dim reportversion\_folder As String

Dim reportversion\_file As String

\*\*\*\*\*

'reopen

flag = 1

If wbdate = Empty Then

wbdate = "2012\_10\_16" & VBA.DateTime.Year(Now()) & "\_" & VBA.DateTime.Month(Now()) & "\_" & VBA.DateTime.Day(Now()) & "2012\_2\_8"

End If

reportversion\_folder = "\_V1-0"

reportversion\_file = "\_V1-0"

folderdescription = ""

\*\*\*\*\*

'Open and activate an Excel workbook (and session)

Dim WbPath As String

Dim WbName As String

Dim wb\_new As Workbook

Dim Wb\_Filtered As Workbook

Dim Wb\_Baseline As Workbook

Dim Wb\_Rz As Workbook

\*\*\*\*\*

\*\*\*\*\*

'Adds the new workbook RD\_MasterDataSet

\*\*\*\*\*

\*\*\*\*\*

'Add a new workbook and move the sheet SmC\_MasterDataSet\_PiLevel to it

If flag Then



```

    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName =
"CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_ForPortfolioReportingAddBaseline_" & wbdate &
reportversion_file & ".xlsb"
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
End If
'Open existing workbook
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_ForPortfolioReportingFilterOut_" &
wbdate & reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_Filtered)
End If
'copy a sheet from another workbook and rename it
If flag Then
    Call z_CopySheetFromWb1ToWb2("SmC_MasterDataset_PiLevel_fldr", Wb_Filtered, wb_new,
"Begin")
'close Wb_GenPart1, closes the active workbook and saves any changes
Wb_Filtered.Close False
'rename 'R3 at the beginning of the name does not work. Error message when adding the pivot
Sheets("SmC_MasterDataset_PiLevel_fldr").name = "Current"
End If

*****

'Rnew - R0
*****

'Open existing workbook
If flag Then
    Call z_ExcelSessionWindowMinimized(wb_new)
    Call z_ExcelSessionWindowNormal(wb_new)
    MsgBox ("check the teamspace path and baseline workbook")
    Stop
    WbPath = "http://teamspace/sites/RD_PMECl/Portfolio%20Reporting/"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_2012_3_5_V1-0.xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_Baseline)
End If
'copy a sheet from another workbook and rename it
If flag Then
    Call z_CopySheetFromWb1ToWb2("R0 SmC_MasterDataSet", Wb_Baseline, wb_new, "Begin")
'close Wb_GenPart1, closes the active workbook and saves any changes
Application.DisplayAlerts = False
Wb_Baseline.Close False
Application.DisplayAlerts = True
'rename 'R0 at the beginning of the name does not work. Error message when adding the pivot
Sheets("R0 SmC_MasterDataset").name = "Baseline"
End If
If flag Then
'in case they have already been deleted in a previous run
On Error Resume Next
    Call z_ShDelete("Sheet1", wb_new)
    Call z_ShDelete("Sheet2", wb_new)

```

```

        Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If

If flag Then
    Call z_OneLinePerPiComparison_ShInNew_ShInOld("Current", "Baseline", _
        "Comp_Current_Baseline", False, False, wbdate)
End If

'*****
'Rnew - Rold
'*****

'Open existing workbook
If flag Then
    Call z_ExcelSessionWindowMinimized(wb_new)
    Call z_ExcelSessionWindowNormal(wb_new)
    MsgBox ("check the teamspace path and Rz workbook, choose the last Rz")
    Stop
    WbPath = "http://teamspace/sites/RD_PMEExcl/Portfolio%20Reporting/"
    WbName =
"CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PortfolioReporting_2012_05_31.xlsx"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_Rz)
End If
    'copy a sheet from another workbook and rename it
If flag Then
    Call z_CopySheetFromWb1ToWb2("R2 SmC_MasterDataSet", Wb_Rz, wb_new, "Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Application.DisplayAlerts = False
    Wb_Rz.Close False
    Application.DisplayAlerts = True
    'rename 'R2 at the beginning of the name does not work. Error message when adding the pivot
    Sheets("R2 SmC_MasterDataset").name = "Retired"
End If
If flag Then
    Call z_WorkbookSave(wb_new)
End If

If flag Then
    Call z_OneLinePerPiComparison_ShInNew_ShInOld("Current", "Retired", _
        "Comp_Current_Retired", True, False, wbdate)
End If

'save and close
If flag Then
    Call z_WorkbookSave(wb_new)
    wb_new.Close True
End If

End Sub

Sub StartAll_Milestones()

```

```

Dim wbdate As String
wbdate = "2012_10_16"
Call m_Main_GenerateMilestoneDataSet(wbdate)
End Sub

Sub m_Main_GenerateMilestoneDataSet(Optional wbdate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
    *****

    Dim flag As Integer
    'Dim wbdate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    If wbdate = Empty Then
        wbdate = "2012_10_16" & VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_" &
        VBA.DateTime.Day(Now()) & "2012_2_8"
    End If
    reportversion_folder = "_V1-0"
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String
    Dim Wb_GenPart1 As Workbook
    Dim wb_new As Workbook
    Dim Wb_OneLinePerPi As Workbook
    *****
    *****

    'Adds the new workbook RD_MasterDataSet
    *****
    *****

    'Add a new workbook and move the MasterDataSet and the PiDataSet into it
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
        reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Milestones_" & wbdate &
        reportversion_file & ".xlsb" 'for test purposes
        Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
    End If
    'Open existing workbook
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
        reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part1_" & wbdate &
        reportversion_file & ".xlsb"
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart1)
    End If
    'copy a sheet from another workbook
    If flag Then

```

```

    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_4", Wb_GenPart1, wb_new,
"Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_GenPart1.Close False
End If
'Open existing workbook
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForOneLinePerId_" &
wbdate & reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_OneLinePerPi)
End If
    'copy a sheet from another workbook
If flag Then
    Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_PiLevel", Wb_OneLinePerPi, wb_new,
"Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_OneLinePerPi.Close False
End If
If flag Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", wb_new)
        Call z_ShDelete("Sheet2", wb_new)
        Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If
    *****

    'write the MS into a new sheet
    Dim FirstYr As Integer
    'FirstYr = 2011
    FirstYr = VBA.DatePart("y", Now()) - 1
If flag Then
    Dim Col_to As Long
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CreateReport", "Begin", "task: create", "file: Milestones_1 and Milestones_2",
CStr(Now()), ""))
    Call z_ShNewFlatValueCopy("PMEC_VBA_MasterDataSet_4",
"SmC_MasterDataSet_Milestones_1", "Begin")
    Col_to = z_GetColumnIndex("ActivityIdentifier", 1, "SmC_MasterDataSet_Milestones_1") + 1
    Call z_InsertEmptyCols("SmC_MasterDataSet_Milestones_1", 1, Col_to)
    Call z_AddColNames(Array("MS belongs to"), "SmC_MasterDataSet_Milestones_1", 1, Col_to)
    Call z_MsBelongsTo("SmC_MasterDataSet_Milestones_1", "Activity Identifier", "MS belongs to")
    Col_to = z_GetColumnIndex("MS belongs to", 1, "SmC_MasterDataSet_Milestones_1")
    Call z_Sort("SmC_MasterDataSet_Milestones_1", Col_to, Col_to)
    Call z_DeleteNotMSRows("SmC_MasterDataSet_Milestones_1", "MS belongs to")
    Dim Col_from As Long
    Col_from = z_GetColumnIndex("MyResourceId", 1, "SmC_MasterDataSet_Milestones_1")
    Col_to = z_GetColumnIndex("Activity Identifier", 1, "SmC_MasterDataSet_Milestones_1")
    Call z_DeleteColumns("SmC_MasterDataSet_Milestones_1", Col_from, Col_to)

```

```

Col_from = z_GetColumnIndex("AC # Trials " & CStr(FirstYr), 1,
"SmC_MasterDataSet_Milestones_1")
Col_to = z_GetColumnIndex("Request status", 1, "SmC_MasterDataSet_Milestones_1")
Call z_DeleteColumns("SmC_MasterDataSet_Milestones_1", Col_from, Col_to)
End If
If flag Then
Call z_ShNewFlatValueCopy("SmC_MasterDataSet_Milestones_1",
"SmC_MasterDataSet_Milestones_2", "Begin")
Col_from = z_GetColumnIndex("Task Customer", 1, "SmC_MasterDataSet_Milestones_2")
Col_to = z_GetColumnIndex("Task Contact", 1, "SmC_MasterDataSet_Milestones_2")
Call z_DeleteColumns("SmC_MasterDataSet_Milestones_2", Col_from, Col_to)
Col_from = z_GetColumnIndex("Task Status", 1, "SmC_MasterDataSet_Milestones_2")
Col_to = z_GetColumnIndex("Duration", 1, "SmC_MasterDataSet_Milestones_2")
Call z_DeleteColumns("SmC_MasterDataSet_Milestones_2", Col_from, Col_to)
Col_from = z_GetColumnIndex("Expected finish export", 1, "SmC_MasterDataSet_Milestones_2")
Col_to = z_GetColumnIndex("Actual finish export", 1, "SmC_MasterDataSet_Milestones_2")
Call z_DeleteColumns("SmC_MasterDataSet_Milestones_2", Col_from, Col_to)
Col_from = z_GetColumnIndex("TkIdentifier", 1, "SmC_MasterDataSet_Milestones_2")
Call z_DeleteColumns("SmC_MasterDataSet_Milestones_2", Col_from, Col_from)
Call z_WorkbookSave(wb_new)
End If
If flag Then
'map attributes from PI report into MS sheet
'read attribute names
Dim Wb_from As Workbook
Dim Rng_from As Range
Dim ColNames_to As Variant
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("Milestones", 2, Wb_from)
ColNames_to = z_ReadAttributeNames_FromSheet_ToArray("Milestones", Rng_from, "ReadACol",
Wb_from)

' ColNames_to = Array( _
' "Portfolio Level 1", "Portfolio Level 2", "Portfolio Level 3", "PI Syngenta Portfolio", "PI Syngenta
Program", "PI Title", "PI Status", "PI Label", "PI Comment", "PI Manager", "PI Sponsor", "PI Stage", "PI
Scope", _
' "PI Type", "PI Sub Type", "PI Investment Category", "PI Customer", "PI Responsibility", "PI
Geography", "PI List of Regions", "PI List of Countries", "PI list of Crops Group", "PI list of Crops", "PI
Purchase Order", "PI Lead AI", "PI List of Active Ingredients", "PI Last Gate Passed", "PI Planned
start", _
' "PI Planned finish", "PI Is confidential ?", _
' "BC Business Case Author", "BC Total NPV", "BC Terminal Value NPV (10% decline)", "BC Sales
Peak", "BC Required", _
' "EAC # SD " & CStr(FirstYr) & "_c", "EAC # SD " & CStr(FirstYr + 1) & "_c", "EAC # SD " &
CStr(FirstYr + 2) & "_c", _
' "EAC # SD " & CStr(FirstYr + 3) & "_c", "EAC # SD " & CStr(FirstYr + 4) & "_c", "EAC # SD_c", "EAC
# Trials " & CStr(FirstYr) & "_c", "EAC # Trials " & CStr(FirstYr + 1) & "_c", "EAC # Trials " & CStr(FirstYr
+ 2) & "_c", "EAC # Trials " & CStr(FirstYr + 3) & "_c", "EAC # Trials " & CStr(FirstYr + 4) & "_c", "EAC #
Trials_c", "EAC Full Costs " & CStr(FirstYr) & "_c", "EAC Full Costs " & CStr(FirstYr + 1) & "_c", "EAC Full
Costs " & CStr(FirstYr + 2) & "_c", _
' "EAC Full Costs " & CStr(FirstYr + 3) & "_c", "EAC Full Costs " & CStr(FirstYr + 4) & "_c", "EAC Full
Costs_c", "EAC SD Full Costs " & CStr(FirstYr) & "_c", "EAC SD Full Costs " & CStr(FirstYr + 1) & "_c",
"EAC SD Full Costs " & CStr(FirstYr + 2) & "_c", "EAC SD Full Costs " & CStr(FirstYr + 3) & "_c", "EAC SD
Full Costs " & CStr(FirstYr + 4) & "_c", "EAC SD Full Costs_c", "EAC Trials Full Costs " & CStr(FirstYr) &

```

```

_c", "EAC Trials Full Costs " & CStr(FirstYr + 1) & "_c", "EAC Trials Full Costs " & CStr(FirstYr + 2) &
_c", _
'   "EAC Trials Full Costs " & CStr(FirstYr + 3) & "_c", "EAC Trials Full Costs " & CStr(FirstYr + 4) &
_c", "EAC Trials Full Costs_c", "EAC Other $ " & CStr(FirstYr) & "_c", "EAC Other $ " & CStr(FirstYr + 1)
& "_c", "EAC Other $ " & CStr(FirstYr + 2) & "_c", "EAC Other $ " & CStr(FirstYr + 3) & "_c", "EAC Other
$ " & CStr(FirstYr + 4) & "_c", "EAC Other $ _c", "EAC Ext $ " & CStr(FirstYr) & "_c", "EAC Ext $ " &
CStr(FirstYr + 1) & "_c", "EAC Ext $ " & CStr(FirstYr + 2) & "_c", _
'   "EAC Ext $ " & CStr(FirstYr + 3) & "_c", "EAC Ext $ " & CStr(FirstYr + 4) & "_c", "EAC Ext $ _c",
"EAC Direct Costs " & CStr(FirstYr) & "_c", "EAC Direct Costs " & CStr(FirstYr + 1) & "_c", "EAC Direct
Costs " & CStr(FirstYr + 2) & "_c", "EAC Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC Direct Costs " &
CStr(FirstYr + 4) & "_c", "EAC Direct Costs_c", "EAC SD Direct Costs " & CStr(FirstYr) & "_c", "EAC SD
Direct Costs " & CStr(FirstYr + 1) & "_c", "EAC SD Direct Costs " & CStr(FirstYr + 2) & "_c", _
'   "EAC SD Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC SD Direct Costs " & CStr(FirstYr + 4) &
_c", "EAC SD Direct Costs_c", "EAC Trials Direct Costs " & CStr(FirstYr) & "_c", "EAC Trials Direct
Costs " & CStr(FirstYr + 1) & "_c", "EAC Trials Direct Costs " & CStr(FirstYr + 2) & "_c", "EAC Trials
Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC Trials Direct Costs " & CStr(FirstYr + 4) & "_c", "EAC
Trials Direct Costs_c", "EAC Other $ Direct Costs " & CStr(FirstYr) & "_c", "EAC Other $ Direct Costs "
& CStr(FirstYr + 1) & "_c", "EAC Other $ Direct Costs " & CStr(FirstYr + 2) & "_c", _
'   "EAC Other $ Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC Other $ Direct Costs " & CStr(FirstYr
+ 4) & "_c", "EAC Other $ Direct Costs_c", "EAC Ext $ Direct Costs " & CStr(FirstYr) & "_c", "EAC Ext $
Direct Costs " & CStr(FirstYr + 1) & "_c", "EAC Ext $ Direct Costs " & CStr(FirstYr + 2) & "_c", "EAC Ext $
Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC Ext $ Direct Costs " & CStr(FirstYr + 4) & "_c", "EAC Ext $
Direct Costs_c" _
'   )
Dim ColStart As Long
ColStart = z_ColSize(1, "SmC_MasterDataSet_Milestones_2") + 1
Call z_AddColNames(ColNames_to, "SmC_MasterDataSet_Milestones_2", 1, ColStart)
Dim ColNames_from As Variant
ColNames_from = ColNames_to

'   ColNames_from = Array( _
'       "Portfolio Level 1", "Portfolio Level 2", "Portfolio Level 3", "PI Syngenta Portfolio", "PI Syngenta
Program", "PI Title", "PI Status", "PI Label", "PI Comment", "PI Manager", "PI Sponsor", "PI Stage", "PI
Scope", _
'       "PI Type", "PI Sub Type", "PI Investment Category", "PI Customer", "PI Responsibility", "PI
Geography", "PI List of Regions", "PI List of Countries", "PI list of Crops Group", "PI list of Crops", "PI
Purchase Order", "PI Lead AI", "PI List of Active Ingredients", "PI Last Gate Passed", "PI Planned
start", _
'       "PI Planned finish", "PI Is confidential ?", _
'       "BC Business Case Author", "BC Total NPV", "BC Terminal Value NPV (10% decline)", "BC Sales
Peak", "BC Required", _
'       "EAC # SD " & CStr(FirstYr) & "_c", "EAC # SD " & CStr(FirstYr + 1) & "_c", "EAC # SD " &
CStr(FirstYr + 2) & "_c", _
'       "EAC # SD " & CStr(FirstYr + 3) & "_c", "EAC # SD " & CStr(FirstYr + 4) & "_c", "EAC # SD_c", "EAC
# Trials " & CStr(FirstYr) & "_c", "EAC # Trials " & CStr(FirstYr + 1) & "_c", "EAC # Trials " & CStr(FirstYr
+ 2) & "_c", "EAC # Trials " & CStr(FirstYr + 3) & "_c", "EAC # Trials " & CStr(FirstYr + 4) & "_c", "EAC #
Trials_c", "EAC Full Costs " & CStr(FirstYr) & "_c", "EAC Full Costs " & CStr(FirstYr + 1) & "_c", "EAC Full
Costs " & CStr(FirstYr + 2) & "_c", _
'       "EAC Full Costs " & CStr(FirstYr + 3) & "_c", "EAC Full Costs " & CStr(FirstYr + 4) & "_c", "EAC Full
Costs_c", "EAC SD Full Costs " & CStr(FirstYr) & "_c", "EAC SD Full Costs " & CStr(FirstYr + 1) & "_c",
"EAC SD Full Costs " & CStr(FirstYr + 2) & "_c", "EAC SD Full Costs " & CStr(FirstYr + 3) & "_c", "EAC SD
Full Costs " & CStr(FirstYr + 4) & "_c", "EAC SD Full Costs_c", "EAC Trials Full Costs " & CStr(FirstYr) &

```

```

"c", "EAC Trials Full Costs " & CStr(FirstYr + 1) & "_c", "EAC Trials Full Costs " & CStr(FirstYr + 2) &
"c", _
'    "EAC Trials Full Costs " & CStr(FirstYr + 3) & "_c", "EAC Trials Full Costs " & CStr(FirstYr + 4) &
"c", "EAC Trials Full Costs_c", "EAC Other $ " & CStr(FirstYr) & "_c", "EAC Other $ " & CStr(FirstYr + 1)
& "_c", "EAC Other $ " & CStr(FirstYr + 2) & "_c", "EAC Other $ " & CStr(FirstYr + 3) & "_c", "EAC Other
$ " & CStr(FirstYr + 4) & "_c", "EAC Other $_c", "EAC Ext $ " & CStr(FirstYr) & "_c", "EAC Ext $ " &
CStr(FirstYr + 1) & "_c", "EAC Ext $ " & CStr(FirstYr + 2) & "_c", _
'    "EAC Ext $ " & CStr(FirstYr + 3) & "_c", "EAC Ext $ " & CStr(FirstYr + 4) & "_c", "EAC Ext $_c",
"EAC Direct Costs " & CStr(FirstYr) & "_c", "EAC Direct Costs " & CStr(FirstYr + 1) & "_c", "EAC Direct
Costs " & CStr(FirstYr + 2) & "_c", "EAC Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC Direct Costs " &
CStr(FirstYr + 4) & "_c", "EAC Direct Costs_c", "EAC SD Direct Costs " & CStr(FirstYr) & "_c", "EAC SD
Direct Costs " & CStr(FirstYr + 1) & "_c", "EAC SD Direct Costs " & CStr(FirstYr + 2) & "_c", _
'    "EAC SD Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC SD Direct Costs " & CStr(FirstYr + 4) &
"c", "EAC SD Direct Costs_c", "EAC Trials Direct Costs " & CStr(FirstYr) & "_c", "EAC Trials Direct
Costs " & CStr(FirstYr + 1) & "_c", "EAC Trials Direct Costs " & CStr(FirstYr + 2) & "_c", "EAC Trials
Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC Trials Direct Costs " & CStr(FirstYr + 4) & "_c", "EAC
Trials Direct Costs_c", "EAC Other $ Direct Costs " & CStr(FirstYr) & "_c", "EAC Other $ Direct Costs "
& CStr(FirstYr + 1) & "_c", "EAC Other $ Direct Costs " & CStr(FirstYr + 2) & "_c", _
'    "EAC Other $ Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC Other $ Direct Costs " & CStr(FirstYr
+ 4) & "_c", "EAC Other $ Direct Costs_c", "EAC Ext $ Direct Costs " & CStr(FirstYr) & "_c", "EAC Ext $
Direct Costs " & CStr(FirstYr + 1) & "_c", "EAC Ext $ Direct Costs " & CStr(FirstYr + 2) & "_c", "EAC Ext $
Direct Costs " & CStr(FirstYr + 3) & "_c", "EAC Ext $ Direct Costs " & CStr(FirstYr + 4) & "_c", "EAC Ext $
Direct Costs_c" _
' )
Dim sSh_from As String
sSh_from = "SmC_MasterDataSet_PiLevel"
Call z_ShMapColumns_FastVersion(sSh_from, "PIdentifier", ColNames_from,
"SmC_MasterDataSet_Milestones_2", "PIdentifier", ColNames_to)
Call z_WorkbookSave(wb_new)
End If
If flag Then
'date attribute planned start
'change the date format
Dim date_Arr As Variant
date_Arr = Array("Planned start", "PI Planned start", "PI Planned finish")
Call z_ChgDateFormat("SmC_MasterDataSet_Milestones_2", date_Arr, 0)
'make date object where not already
Dim Rng As Range
Dim Date_ColNames As Variant
Dim Date_Col_i As Long
Dim Date_ColName_i As String
Dim Pild_Col As Long
Dim RowSize As Long
Pild_Col = z_GetColumnIndex("PIdentifier", 1, "SmC_MasterDataSet_Milestones_2")
RowSize = z_RowSize(Pild_Col, "SmC_MasterDataSet_Milestones_2")
Date_ColNames = Array("Planned start")
For iter = LBound(Date_ColNames) To UBound(Date_ColNames)
    Date_ColName_i = Date_ColNames(iter)
    Date_Col_i = z_GetColumnIndex(Date_ColName_i, 1, "SmC_MasterDataSet_Milestones_2")
    Set Rng = Range(Cells(1, Date_Col_i), Cells(RowSize, Date_Col_i))
    Call z_DateCorrection(Rng)
Next iter
'rename attributes

```

```

    Call z_RenameAttribute("SmC_MasterDataSet_Milestones_2", 1, "Syngenta Portfolio", "Activity
Syngenta Portfolio")
    Call z_RenameAttribute("SmC_MasterDataSet_Milestones_2", 1, "Task Title", "TK Task Title")
    Call z_RenameAttribute("SmC_MasterDataSet_Milestones_2", 1, "Description", "Activity
Description")
    Call z_RenameAttribute("SmC_MasterDataSet_Milestones_2", 1, "Task Location", "TK Task
Location")
    Call z_RenameAttribute("SmC_MasterDataSet_Milestones_2", 1, "Planned start", "Activity Planned
Start")
End If

```

```

'add timestamps
Dim StampDate As String
StampDate = z_StampDate(wbdate)
Call z_AddTimeStamp("SmC_MasterDataSet_Milestones_2", StampDate, 1, 1, 25)

Call z_WorkbookSave(wb_new)
wb_new.Close True

```

```
End Sub
```

```

Sub StartAll_AllIds()
    Dim wbdate As String
    wbdate = "2012_10_16"
    Call m_Main_AllIds_FindRemovedOnes(wbdate)
End Sub

```

```

Sub m_Main_AllIds_FindRemovedOnes(Optional wbdate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
    *****

    Dim flag As Integer
    'Dim wbdate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    flag_OpenExistingWbs = 1
    If wbdate = Empty Then
        wbdate = "2012_10_16" & VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) & "2012_2_8"
    End If
    reportversion_folder = "_V1-0"
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String

```



```

Dim Wb_GenPart1 As Workbook
Dim wb_new As Workbook
Dim Wb_OneLinePerPi As Workbook
*****

*****

'Adds the new workbook RD_MasterDataSet
*****

*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_AllIds_" & wdate &
reportversion_file & ".xlsb" 'for test purposes
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
End If
'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Generation_Part1_" & wdate &
reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_GenPart1)
End If
'copy a sheet from another workbook
If flag_OpenExistingWbs Then
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_MasterDataSet_4", Wb_GenPart1, wb_new,
"Begin")
    Call z_CopySheetFromWb1ToWb2("PMEC_VBA_PiDataSet_2", Wb_GenPart1, wb_new, "Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_GenPart1.Close False
End If
'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForOneLinePerId_" &
wdate & reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_OneLinePerPi)
End If
'copy a sheet from another workbook
If flag_OpenExistingWbs Then
    Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_PiLevel", Wb_OneLinePerPi, wb_new,
"Begin")
    Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_TkLevel", Wb_OneLinePerPi, wb_new,
"Begin")
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_OneLinePerPi.Close False
End If
If flag_OpenExistingWbs Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
    Call z_ShDelete("Sheet1", wb_new)

```

```

        Call z_ShDelete("Sheet2", wb_new)
        Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If
If flag_OpenExistingWbs Then
    Call z_ShNew("PMEC_VBA_AllPis", "Begin")
    Call z_ShNew("PMEC_VBA_AllTks", "Begin")
    Call z_ShNew("RemovedPis_(OneLinePrPi-Empty)", "Begin")
    Call z_ShNew("RemovedTks_(OneLinePrTk-Empty)", "Begin")
End If
'*****
'Make sheets "PMEC_VBA_AllPis" and "PMEC_VBA_AllTks"
'Determine HasResource Yes/No
If flag Then
    Dim col_Pild_from As Long
    Dim col_Tkld_from As Long
    Dim Col_MyResld_from As Long
    col_Pild_from = z_GetColumnIndex("Pildentifier", 1, "PMEC_VBA_MasterDataSet_4")
    col_Tkld_from = z_GetColumnIndex("Tkldentifier", 1, "PMEC_VBA_MasterDataSet_4")
    Col_MyResld_from = z_GetColumnIndex("MyResourceId", 1, "PMEC_VBA_MasterDataSet_4")
    Dim Col_Pild_to As Long
    Dim Col_PiHasRes_to As Long
    Dim Col_Tkld_to As Long
    Dim Col_TkHasRes_to As Long
    Dim Col_Rsld_to As Long
    Col_Pild_to = 1
    Col_PiHasRes_to = 2
    Col_Tkld_to = 3
    Col_TkHasRes_to = 4
    Col_Rsld_to = 5
    Sheets("PMEC_VBA_AllPis").Activate
    Call z_AddColNames(Array("Pildentifier", "Pi has Resource(s)"), "PMEC_VBA_AllPis", 1, 1)
    Sheets("PMEC_VBA_AllTks").Activate
    Call z_AddColNames(Array("Pildentifier", "Pi has Resource(s)", "Tkldentifier", "Task has
Resource(s)"), "PMEC_VBA_AllTks", 1, 1)
    Dim Cell_Pild_from As Range
    Dim Rng_from As Range
    Dim RowSize As Long
    Sheets("PMEC_VBA_MasterDataSet_4").Activate
    RowSize = z_RowSize(col_Pild_from, "PMEC_VBA_MasterDataSet_4")
    Set Rng_from = Range(Cells(2, col_Pild_from), Cells(RowSize, col_Pild_from))
    Dim Row_to As Long
    Dim Resource_found As Boolean
    'Pis
    Sheets("PMEC_VBA_MasterDataSet_4").Activate
    Resource_found = False
    Row_to = 2
    For Each Cell_Pild_from In Rng_from
        'go on and check
        If Cell_Pild_from.Value = Cell_Pild_from.Offset(1, 0).Value Then
            'if Resource found set flag to 1
            If Cells(Cell_Pild_from.Row, Col_MyResld_from) <> Empty Then

```

```

        Resource_found = True
    End If
    'last row of same Pi
Else
    'if Resource found set flag to 1
    If Cells(Cell_Pild_from.Row, Col_MyResId_from) <> Empty Then
        Resource_found = True
    End If
    'write Pi and PiRs and set flag back to 0
    Sheets("PMEC_VBA_AllPis").Cells(Row_to, Col_Pild_to).Value = Cell_Pild_from.Value
    Sheets("PMEC_VBA_AllPis").Cells(Row_to, Col_PiHasRes_to).Value = Resource_found
    Resource_found = False
    Row_to = Row_to + 1
End If
Next
'Tks
Sheets("PMEC_VBA_MasterDataSet_4").Activate
Set Rng_from = Range(Cells(2, col_TkId_from), Cells(RowSize, col_TkId_from))
Resource_found = False
Row_to = 2
For Each Cell_TkId_from In Rng_from
    If Cell_TkId_from <> Empty Then
        'go on and check
        If Cell_TkId_from.Value = Cell_TkId_from.Offset(1, 0).Value Then
            'if Resource found set flag to 1
            If Cells(Cell_TkId_from.Row, Col_MyResId_from) <> Empty Then
                Resource_found = True
            End If
            'last row of same Pi
        Else
            'if Resource found set flag to 1
            If Cells(Cell_TkId_from.Row, Col_MyResId_from) <> Empty Then
                Resource_found = True
            End If
            'write Pi and PiRs and set flag back to 0
            Sheets("PMEC_VBA_AllTks").Cells(Row_to, Col_Pild_to).Value = Cells(Cell_TkId_from.Row,
col_Pild_from)
            Sheets("PMEC_VBA_AllTks").Cells(Row_to, Col_TkId_to).Value = Cell_TkId_from.Value
            Sheets("PMEC_VBA_AllTks").Cells(Row_to, Col_TkHasRes_to).Value = Resource_found
            Resource_found = False
            Row_to = Row_to + 1
        End If
    End If
Next
End If
End If
*****

'Make sheets "RemovedPis_(OneLinePrPi-Empty)" and "RemovedTks_(OneLinePrTk-Empty)"
'those with "Pildentifier from SmC_MasterDataSet_PiLevel" or "TkIdentifier from
SmC_MasterDataSet_TkLevel" equal empty
If flag Then
    'PI: map identifier
    Dim ColNames_to As Variant
    ColNames_to = Array( _

```

```

        "PiIdentifier from SmC_MasterDataSet_PiLevel")
Dim ColStart As Long
ColStart = z_ColSize(1, "PMEC_VBA_AllPis") + 1
Call z_AddColNames(ColNames_to, "PMEC_VBA_AllPis", 1, ColStart)
Dim ColNames_from As Variant
ColNames_from = Array( _
    "PiIdentifier")
Dim sSh_from As String
sSh_from = "SmC_MasterDataSet_PiLevel"
Call z_ShMapColumns_FastVersion(sSh_from, "PiIdentifier", ColNames_from, "PMEC_VBA_AllPis",
"PiIdentifier", ColNames_to)
Call z_WorkbookSave(wb_new)
'PI: filter and copy paste
Sheets("PMEC_VBA_AllPis").Activate
Col_from = 1
Col_to = 3
col_Pild_from = 1
RowSize = z_RowSize(col_Pild_from, "PMEC_VBA_AllPis")
Set Rng_from = Range(Cells(2, Col_from), Cells(RowSize, Col_to))
Call z_SetFilter("PMEC_VBA_AllPis", Rng_from, "PiIdentifier from SmC_MasterDataSet_PiLevel",
Array("="))
Call z_Copy_AutoFilterRange("PMEC_VBA_AllPis", Rng_from, "RemovedPis_(OneLinePrPi-Empty)",
Sheets("RemovedPis_(OneLinePrPi-Empty)").Cells(2, 1))
Call z_CopyRow("PMEC_VBA_AllPis", 1, "RemovedPis_(OneLinePrPi-Empty)", 1)

'TK: map identifier
ColNames_to = Array( _
    "TkIdentifier from SmC_MasterDataSet_TkLevel")
ColStart = z_ColSize(1, "PMEC_VBA_AllTks") + 1
Call z_AddColNames(ColNames_to, "PMEC_VBA_AllTks", 1, ColStart)
ColNames_from = Array( _
    "TkIdentifier")
sSh_from = "SmC_MasterDataSet_TkLevel"
Call z_ShMapColumns_FastVersion(sSh_from, "TkIdentifier", ColNames_from,
"PMEC_VBA_AllTks", "TkIdentifier", ColNames_to)
Call z_WorkbookSave(wb_new)
'TK: filter and copy paste
Sheets("PMEC_VBA_AllTks").Activate
Col_from = 1
Col_to = 5
col_Pild_from = 1
RowSize = z_RowSize(col_Pild_from, "PMEC_VBA_AllTks")
Set Rng_from = Range(Cells(2, Col_from), Cells(RowSize, Col_to))
Call z_SetFilter("PMEC_VBA_AllTks", Rng_from, "TkIdentifier from SmC_MasterDataSet_TkLevel",
Array("="))
Call z_Copy_AutoFilterRange("PMEC_VBA_AllTks", Rng_from, "RemovedTks_(OneLinePrTk-
Empty)", Sheets("RemovedTks_(OneLinePrTk-Empty)").Cells(2, 1))
Call z_CopyRow("PMEC_VBA_AllTks", 1, "RemovedTks_(OneLinePrTk-Empty)", 1)
End If
*****

'Fill sheet "RemovedPis_(OneLinePrPi-Empty)"
'map columns in ColNames_to and make some formatting
If flag Then

```

```

'Mapping only on the Ids removed from the SmC_PMEC_VBA_MasterDataSet
'map attributes
Dim Wb_from As Workbook
'Dim Rng_from As Range
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 2, Wb_from)
ColNames_to = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from, "ReadACol",
Wb_from)

ColStart = z_ColSize(1, "RemovedPis_(OneLinePrPi-Empty)") + 1
Call z_AddColNames(ColNames_to, "RemovedPis_(OneLinePrPi-Empty)", 1, ColStart)

Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 3, Wb_from)
ColNames_from = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from, "ReadACol",
Wb_from)

sSh_from = "PMEC_VBA_PiDataSet_2"
Call z_ShMapColumns_FastVersion(sSh_from, "PI Identifier", ColNames_from,
"RemovedPis_(OneLinePrPi-Empty)", "PIdentifier", ColNames_to)
'for value attributes:replace all blanks with "0"
'for non value attributes: replace all blanks with "(blank)"
Call z_replaceEmptyCellsWithZero("RemovedPis_(OneLinePrPi-Empty)", "Like", "EAC*")
Call z_replaceEmptyCellsWithBLANK("RemovedPis_(OneLinePrPi-Empty)", "NotLike", "EAC*")
'Date Columns
date_Arr = Array("PI Planned start", "PI Planned finish")
Call z_ChgDateFormat("RemovedPis_(OneLinePrPi-Empty)", date_Arr, 0)
Call z_DateCorrection2("RemovedPis_(OneLinePrPi-Empty)", "PIdentifier", date_Arr)
Call z_WorkbookSave(wb_new)
End If
'*****

'Adapt sheet "PMEC_VBA_MasterDataSet_4"
'Calculate EAC Full Cost
If flag Then
'calculate EAC
Sheets("PMEC_VBA_MasterDataSet_4").Activate
RowSize = z_RowSize(col_Pild_from, "PMEC_VBA_MasterDataSet_4")
Dim Col_Activity_Id As Long
Dim Col_EAC As Long
Dim Col_AC As Long
Dim Col_ETC As Long
Col_Activity_Id = z_GetColumnIndex("Activity Identifier", 1, "PMEC_VBA_MasterDataSet_4")
Col_EAC = z_GetColumnIndex("EAC Full Costs_c", 1, "PMEC_VBA_MasterDataSet_4")
Col_AC = z_GetColumnIndex("AC Full Costs", 1, "PMEC_VBA_MasterDataSet_4")
Col_ETC = z_GetColumnIndex("ETC Full Costs", 1, "PMEC_VBA_MasterDataSet_4")
Call z_ChgFmt_CostCols("PMEC_VBA_MasterDataSet_4", 69, 69) 'Col_ETC, Col_ETC)
Call z_ChgFmt_CostCols("PMEC_VBA_MasterDataSet_4", 63, 63) 'Col_AC, Col_AC)
Set Rng_from = Range(Cells(2, col_Pild_from), Cells(RowSize, col_Pild_from))
For Each Cell_Pild_from In Rng_from
If left(Cells(Cell_Pild_from.Row, Col_Activity_Id).Value, 2) = "TK" Then
Cells(Cell_Pild_from.Row, Col_EAC) = Cells(Cell_Pild_from.Row, Col_ETC) +
Cells(Cell_Pild_from.Row, Col_AC)
End If
Next
End If

```

```

*****

'Fill sheet "RemovedTks_(OneLinePrTk-Empty)"
'add column names in ColNames_to to the sheet "RemovedTks_(OneLinePrTk-Empty)"
'map columns in ColNames_to from "PMEC_VBA_MasterDataSet_4"
'map columns in ColNames_to from "PMEC_VBA_PiDataSet_2"
'and make some formatting
If flag Then
    'add col names to the sheet "RemovedTks_(OneLinePrTk-Empty)"
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 4, Wb_from)
    ColNames_to = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from, "ReadACol",
Wb_from)

    ColStart = z_ColSize(1, "RemovedTks_(OneLinePrTk-Empty)") + 1
    Call z_AddColNames(ColNames_to, "RemovedTks_(OneLinePrTk-Empty)", 1, ColStart)
    'map Attributes from "PMEC_VBA_MasterDataSet_4"
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 5, Wb_from)
    ColNames_to = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from, "ReadACol",
Wb_from)

    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 6, Wb_from)
    ColNames_from = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from, "ReadACol",
Wb_from)

    sSh_from = "PMEC_VBA_MasterDataSet_4"
    Call z_ShMapColumns_FastVersion(sSh_from, "Activity Identifier", ColNames_from,
"RemovedTks_(OneLinePrTk-Empty)", "TkIdentifier", ColNames_to)
    'map Attributes from "PMEC_VBA_PiDataSet_2"
    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 7, Wb_from)
    ColNames_to = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from, "ReadACol",
Wb_from)

    Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 8, Wb_from)
    ColNames_from = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from, "ReadACol",
Wb_from)

    sSh_from = "PMEC_VBA_PiDataSet_2"
    Call z_ShMapColumns_FastVersion(sSh_from, "PI Identifier", ColNames_from,
"RemovedTks_(OneLinePrTk-Empty)", "PiIdentifier", ColNames_to)
    'for value attributes:replace all blanks with "0"
    'for non value attributes: replace all blanks with "(blank)"
    Call z_replaceEmptyCellsWithZero("RemovedTks_(OneLinePrTk-Empty)", "Like", "EAC*")
    Call z_replaceEmptyCellsWithBLANK("RemovedTks_(OneLinePrTk-Empty)", "NotLike", "EAC*")
    'Date Columns
    date_Arr = Array("PI Planned start", "PI Planned finish", "Activity Planned Start", "Activity Expected
finish", "Activity Actual Start", "Activity Actual Finish")
    Call z_ChgDateFormat("RemovedTks_(OneLinePrTk-Empty)", date_Arr, 0)
    Call z_DateCorrection2("RemovedTks_(OneLinePrTk-Empty)", "PiIdentifier", date_Arr)
    Call z_WorkbookSave(wb_new)
End If
*****

'make sheet "SmC_FullDataSet_PiLevel"
'copy columns ColNames_SmC_MasterDataSet_PiLevel
'from sheet "SmC_MasterDataSet_PiLevel" into sheet "SmC_FullDataSet_PiLevel"

```

```

'copy columns ColNames_Removed_Pis
'from sheet "RemovedPis_(OneLinePrPi-Empty)" into sheet "SmC_FullDataSet_PiLevel"
If flag Then
'Generate the sheet with all Pis
Call z_ShNew("SmC_FullDataSet_PiLevel", "Begin")
'copy columns from sheet "SmC_MasterDataSet_PiLevel"
Dim ColNames_SmC_MasterDataSet_PiLevel As Variant
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 9, Wb_from)
ColNames_SmC_MasterDataSet_PiLevel = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs",
Rng_from, "ReadACol", Wb_from)

Call z_AddColNames(ColNames_SmC_MasterDataSet_PiLevel, "SmC_FullDataSet_PiLevel", 1, 1)
Call z_CopyRange_FromColumnArray("SmC_MasterDataSet_PiLevel", "SmC_FullDataSet_PiLevel",
ColNames_SmC_MasterDataSet_PiLevel, _
    "PiIdentifier", 250)
'copy columns from sheet "RemovedPis_(OneLinePrPi-Empty)"
Dim ColNames_Removed_Pis As Variant
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 10, Wb_from)
ColNames_Removed_Pis = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from,
"ReadACol", Wb_from)

Call z_AddColNames(Array("PiIdentifier", "Pi has Resource(s)", "PiIdentifier from
SmC_MasterDataSet_PiLevel"), "SmC_FullDataSet_PiLevel", 1, 1)
Call z_CopyRange_FromColumnArray("RemovedPis_(OneLinePrPi-Empty)",
"SmC_FullDataSet_PiLevel", ColNames_Removed_Pis, _
    "PiIdentifier", 250)
'for value attributes:replace all blanks with "0"
'for non value attributes: replace all blanks with "(blank)"
Call z_replaceEmptyCellsWithZero("SmC_FullDataSet_PiLevel", "Like", "EAC*")
Call z_replaceEmptyCellsWithBLANK("SmC_FullDataSet_PiLevel", "NotLike", "EAC*")
'change format for cost columns 0 -> 0.00
Dim FirstYr As Integer
FirstYr = VBA.DateTime.year(Now()) - 1
Dim FromCol As Long
Dim ToCol As Long
FromCol = z_GetColumnIndex("EAC # SD " & CStr(FirstYr) & "_c", 1, "SmC_FullDataSet_PiLevel")
ToCol = z_GetColumnIndex("EAC Ext $ Direct Costs" & "_c", 1, "SmC_FullDataSet_PiLevel")
ToCol = z_GetColumnIndex("EAC Direct Costs" & "_c", 1, "SmC_FullDataSet_PiLevel")
Call z_ChgFmt_CostCols("SmC_FullDataSet_PiLevel", FromCol, ToCol, "###0.00")
'Date Columns
date_Arr = Array("PI Planned start", "PI Planned finish", "BC First year of sales")
Call z_ChgDateFormat("SmC_FullDataSet_PiLevel", date_Arr, 0)
'Call z_DateCorrection2("SmC_FullDataSet_PiLevel", "PiIdentifier", date_Arr)
End If
*****
'make sheet "SmC_FullDataSet_TkLevel"
'copy columns ColNames_SmC_MasterDataSet_TkLevel
'from sheet "SmC_MasterDataSet_TkLevel" into sheet "SmC_FullDataSet_TkLevel"
'copy columns ColNames_Removed_Pis
'from sheet "RemovedTks_(OneLinePrTk-Empty)" into sheet "SmC_FullDataSet_TkLevel"
'for non value attributes and especially "TK Customer" and "List of task customer": replace all
blanks with "(blank)"
If flag Then

```

```

'Generate the sheet with all TKs
Call z_ShNew("SmC_FullDataSet_TkLevel", "Begin")
'copy columns from sheet "SmC_MasterDataSet_TkLevel"
Dim ColNames_SmC_MasterDataSet_TkLevel As Variant
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 11, Wb_from)
ColNames_SmC_MasterDataSet_TkLevel = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs",
Rng_from, "ReadACol", Wb_from)

Call z_CopyRange_FromColumnArray("SmC_MasterDataSet_TkLevel", "SmC_FullDataSet_TkLevel",
ColNames_SmC_MasterDataSet_TkLevel, _
    "TkIdentifier", 250)
Call z_AddColNames(ColNames_SmC_MasterDataSet_TkLevel, "SmC_FullDataSet_TkLevel", 1, 1)
'copy columns from sheet "RemovedTks_(OneLinePrTk-Empty)" below
Dim ColNames_Removed_Tks As Variant
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 12, Wb_from)
ColNames_Removed_Tks = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from,
"ReadACol", Wb_from)

Call z_CopyRange_FromColumnArray("RemovedTks_(OneLinePrTk-Empty)",
"SmC_FullDataSet_TkLevel", ColNames_Removed_Tks, _
    "TkIdentifier", 250)
Dim ColNames_Removed2_Tks As Variant
Set Rng_from = z_RangeOfWb_AttributeNamesFromSheetToArray("ALL_IDs", 13, Wb_from)
ColNames_Removed2_Tks = z_ReadAttributeNames_FromSheet_ToArray("ALL_IDs", Rng_from,
"ReadACol", Wb_from)

Call z_AddColNames(ColNames_Removed2_Tks, "SmC_FullDataSet_TkLevel", 1, 1, , "No")

'for value attributes:replace all blanks with "0"
'for non value attributes: replace all blanks with "(blank)"
Call z_replaceEmptyCellsWithZero("SmC_FullDataSet_TkLevel", "Like", "EAC*")
Call z_replaceEmptyCellsWithBLANK("SmC_FullDataSet_TkLevel", "NotLike", "EAC*")
FromCol = z_GetColumnIndex("EAC # SD " & CStr(FirstYr) & "_c", 1, "SmC_FullDataSet_TkLevel")
ToCol = z_GetColumnIndex("EAC Ext $ Direct Costs" & "_c", 1, "SmC_FullDataSet_TkLevel")
ToCol = z_GetColumnIndex("EAC Direct Costs" & "_c", 1, "SmC_FullDataSet_TkLevel")
Call z_ChgFmt_CostCols("SmC_FullDataSet_TkLevel", FromCol, ToCol, "###0.00")
'Date Columns
date_Arr = Array("PI Planned start", "PI Planned finish", "Activity Planned Start", "Activity Expected
finish", "Activity Actual Start", "Activity Actual Finish", "BC First year of sales")
Call z_ChgDateFormat("SmC_FullDataSet_TkLevel", date_Arr, 0)
'Call z_DateCorrection2("SmC_FullDataSet_TkLevel", "PIdentifier", date_Arr)
End If
Call z_WorkbookSave(wb_new)
wb_new.Close False

End Sub

Sub StartAll_CheckBusinessRules()
Dim wbdate As String
wbdate = "2012_10_16"
Call m_Main_CheckBusinessRules(wbdate)
End Sub

```



```

Sub m_Main_CheckBusinessRules(Optional wdate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
    *****

    Dim flag As Integer
    Dim flag_OpenWbAllIds As Integer
    'Dim wdate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    flag_OpenExistingWbs = 1
    If wdate = Empty Then
        wdate = "2012_10_16" & VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_" & VBA.DateTime.Day(Now()) & "2012_2_8"
    End If
    reportversion_folder = "_V1-0"
    reportversion_file = "_V1-0"
    folderdescription = ""
    *****

    'Open and activate an Excel workbook (and session)
    Dim WbPath As String
    Dim WbName As String
    Dim Wb_AllIds As Workbook
    Dim Wb_OneLinePerId2 As Workbook
    Dim wb_new As Workbook
    Dim Wb_MasterPlan As Workbook
    *****
    *****

    'Adds the new workbook RD_MasterDataSet
    *****
    *****

    'Add a new workbook and move the MasterDataSet and the PiDataSet into it
    If flag Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
            reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_QualityCheckBusinessRules_" &
            wdate & reportversion_file & ".xlsb" 'for test purposes
        Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
    End If
    'Open existing workbook
    If flag_OpenExistingWbs Then
        WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
            reportversion_folder & folderdescription & "\"
        WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_AllIds_" & wdate &
            reportversion_file & ".xlsb"
        Call z_OpenAndActivateWb(WbName, WbPath, Wb_AllIds)
    End If
    'copy a sheet from another workbook
    If flag_OpenExistingWbs Then
        'Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_PiLevel", Wb_AllIds, Wb_New, "Begin")
    End If

```

```

'Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_TkLevel", Wb_AllIds, Wb_New, "Begin")
'Call z_CopySheetFromWb1ToWb2("RemovedPis_(OneLinePrPi-Empty)", Wb_AllIds, Wb_New,
"Begin")
'Call z_CopySheetFromWb1ToWb2("RemovedTks_(OneLinePrTk-Empty)", Wb_AllIds, Wb_New,
"Begin")
Call z_CopySheetFromWb1ToWb2("SmC_FullDataSet_PiLevel", Wb_AllIds, wb_new, "Begin")
Call z_CopySheetFromWb1ToWb2("SmC_FullDataSet_TkLevel", Wb_AllIds, wb_new, "Begin")
Wb_AllIds.Close False
End If
'Open existing workbook
If flag_OpenExistingWbs Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForOneLinePerId2_" &
wbdate & reportversion_file & ".xlsb"
Call z_OpenAndActivateWb(WbName, WbPath, Wb_OneLinePerId2)
End If
'copy a sheet from another workbook
If flag_OpenExistingWbs Then
Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_ResourceLevel", Wb_OneLinePerId2,
wb_new, "Begin")
Wb_OneLinePerId2.Close False
End If
'Open existing workbook from teamspace without check out
If flag_OpenExistingWbs Then
WbPath = "http://teamspace/sites/PMECTeam/Restricted_Documents/"
WbName = "MasterPlan - SmC Data Quality Rules" & ".xlsx"
Call z_OpenAndActivateWb_fromSharepoint_CheckOutOrReadOnly(WbPath, WbName,
"ReadOnly", Wb_MasterPlan)
End If
'copy a sheet from another workbook
If flag_OpenExistingWbs Then
Call z_CopySheetFromWb1ToWb2("Master Plan", Wb_MasterPlan, wb_new, "Begin")
'close Wb_GenPart1, closes the active workbook and saves any changes
Wb_MasterPlan.Close False
End If

If flag_OpenExistingWbs Then
'in case they have already been deleted in a previous run
On Error Resume Next
Call z_ShDelete("Sheet1", wb_new)
Call z_ShDelete("Sheet2", wb_new)
Call z_ShDelete("Sheet3", wb_new)
On Error GoTo 0
Call z_WorkbookSave(wb_new)
End If

'Input Sheets
If flag Then
Call z_AutoFilterOn("SmC_FullDataSet_PiLevel", 1)
'get the row size
Dim Pild_Col As Long
Pild_Col = z_GetColumnIndex("Pildentifier", 1, "SmC_FullDataSet_PiLevel")

```

```

Dim RowSize As Long
Dim ColSize As Long
RowSize = z_RowSize(Pild_Col, "SmC_FullDataSet_PiLevel")
ColSize = z_ColSize(1, "SmC_FullDataSet_PiLevel")
'set the range
Dim Rng_PiLev As Range
Sheets("SmC_FullDataSet_PiLevel").Select
'check if time stamp is there
If Cells(1, 1).Value Like "Data as of*" Or Cells(2, 1).Value Like "Data as of*" Then
    Set Rng_PiLev = Range(Cells(2, 2), Cells(RowSize, ColSize))
Else
    Set Rng_PiLev = Range(Cells(2, 1), Cells(RowSize, ColSize))
End If
Rng_PiLev.Select
Selection.Interior.Color = xlNone 'RGB(365, 365, 365)
End If
If flag_OpenExistingWbs Then
    Call z_AutofilterOff("SmC_FullDataSet_TkLevel", 1) 'Call
z_AutofilterOn("SmC_FullDataSet_TkLevel", 1)
    Call z_AutofilterOff("SmC_MasterDataSet_ResourceLevel", 1) 'Call
z_AutofilterOn("SmC_MasterDataSet_ResourceLevel", 1)
    'add column"EAC CurrentYear+n" and "EAC NextYear+n"
    Dim FirstYr As Integer
    FirstYr = VBA.DateTimes.year(Now()) - 1 'FirstYr = 2011
    'EAC CurrentYear + n -> EAC Full Costs Current Year & Future Years
    'EAC NextYear + n -> EAC Full Costs Future Years (Excl. Current Year)
    Call z_AddAuxiliaryAttribute_Sum("SmC_FullDataSet_TkLevel", "PIdentifier", "EAC Full Costs
Current Year & Future Years", _
        Array("EAC Full Costs " & CStr(FirstYr + 1) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 4) & "_c"))
    Call z_AddAuxiliaryAttribute_Sum("SmC_FullDataSet_TkLevel", "PIdentifier", "EAC Full Costs
Future Years (Excl. Current Year)", _
        Array("EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 4) & "_c"))
    Call z_AddAuxiliaryAttribute_Sum("SmC_MasterDataSet_ResourceLevel", "PIdentifier", "EAC Full
Costs Current Year & Future Years", _
        Array("EAC Full Costs " & CStr(FirstYr + 1) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 4) & "_c"))
    Call z_AddAuxiliaryAttribute_Sum("SmC_MasterDataSet_ResourceLevel", "PIdentifier", "EAC Full
Costs Future Years (Excl. Current Year)", _
        Array("EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 4) & "_c"))
End If
If flag Then
    'get the row size
    Dim TkId_Col As Long
    TkId_Col = z_GetColumnIndex("TkIdentifier", 1, "SmC_FullDataSet_TkLevel")

```

```

RowSize = z_RowSize(TkId_Col, "SmC_FullDataSet_TkLevel")
ColSize = z_ColSize(1, "SmC_FullDataSet_TkLevel")
'set the range
Dim Rng_TkLev As Range
Sheets("SmC_FullDataSet_TkLevel").Select
If Cells(1, 1).Value Like "Data as of*" Or Cells(2, 1).Value Like "Data as of*" Then
    Set Rng_TkLev = Range(Cells(2, 2), Cells(RowSize, ColSize))
Else
    Set Rng_TkLev = Range(Cells(2, 1), Cells(RowSize, ColSize))
End If
Rng_TkLev.Select
Selection.Interior.Color = xlNone 'RGB(365, 365, 365)
End If
If flag Then
    Call z_AutofilterOn("SmC_MasterDataSet_ResourceLevel", 1)
    'get the row size
    Dim RsId_Col As Long
    RsId_Col = z_GetColumnIndex("RsIdentifier", 1, "SmC_MasterDataSet_ResourceLevel")
    RowSize = z_RowSize(RsId_Col, "SmC_MasterDataSet_ResourceLevel")
    ColSize = z_ColSize(1, "SmC_MasterDataSet_ResourceLevel")
    'set the range
    Dim Rng_RsLev As Range
    Sheets("SmC_MasterDataSet_ResourceLevel").Select
    If Cells(1, 1).Value Like "Data as of*" Or Cells(2, 1).Value Like "Data as of*" Then
        Set Rng_RsLev = Range(Cells(2, 2), Cells(RowSize, ColSize))
    Else
        Set Rng_RsLev = Range(Cells(2, 1), Cells(RowSize, ColSize))
    End If
    Rng_RsLev.Select
    Selection.Interior.Color = xlNone 'RGB(365, 365, 365)
End If

'Output Sheets
If flag Then
    'add sheet
    Call z_ShNew("QualityChecksPI_FoundErrors", "Begin", , , "Delete")
    'add col names
    Call z_AddColNames(Array("BusinessRuleNr"), "QualityChecksPI_FoundErrors", 1, 1)
    Sheets("SmC_FullDataSet_PiLevel").Select
    ColSize = z_ColSize(1, "SmC_FullDataSet_PiLevel")
    Dim Rng_PiLev_ColNames As Range
    Sheets("SmC_FullDataSet_PiLevel").Select
    'check if time stamp is there
    If Cells(1, 1).Value Like "Data as of*" Or Cells(2, 1).Value Like "Data as of*" Then
        Set Rng_PiLev_ColNames = Range(Cells(1, 2), Cells(1, ColSize))
    Else
        Set Rng_PiLev_ColNames = Range(Cells(1, 1), Cells(1, ColSize))
    End If
    Call z_CopyRange("All", "SmC_FullDataSet_PiLevel", Rng_PiLev_ColNames,
"QualityChecksPI_FoundErrors", Sheets("QualityChecksPI_FoundErrors").Cells(1, 2))
End If
If flag Then
    'add sheet

```

```

Call z_ShNew("QualityChecksTK_FoundErrors", "Begin", , , "Delete")
'add col names
Call z_AddColNames(Array("BusinessRuleNr"), "QualityChecksTK_FoundErrors", 1, 1)
Sheets("SmC_FullDataSet_TkLevel").Select
ColSize = z_ColSize(1, "SmC_FullDataSet_TkLevel")
Dim Rng_TkLev_ColNames As Range
Sheets("SmC_FullDataSet_TkLevel").Select
'check if time stamp is there
If Cells(1, 1).Value Like "Data as of*" Or Cells(2, 1).Value Like "Data as of*" Then
    Set Rng_TkLev_ColNames = Range(Cells(1, 2), Cells(1, ColSize))
Else
    Set Rng_TkLev_ColNames = Range(Cells(1, 1), Cells(1, ColSize))
End If
Call z_CopyRange("All", "SmC_FullDataSet_TkLevel", Rng_TkLev_ColNames,
"QualityChecksTK_FoundErrors", Sheets("QualityChecksTK_FoundErrors").Cells(1, 2))
End If
If flag Then
'add sheet
Call z_ShNew("QualityChecksRS_FoundErrors", "Begin", , , "Delete")
'add col names
Call z_AddColNames(Array("BusinessRuleNr"), "QualityChecksRS_FoundErrors", 1, 1)
Sheets("SmC_MasterDataSet_ResourceLevel").Select
ColSize = z_ColSize(1, "SmC_MasterDataSet_ResourceLevel")
Dim Rng_RsLev_ColNames As Range
Sheets("SmC_MasterDataSet_ResourceLevel").Select
'check if time stamp is there
If Cells(1, 1).Value Like "Data as of*" Or Cells(2, 1).Value Like "Data as of*" Then
    Set Rng_RsLev_ColNames = Range(Cells(1, 2), Cells(1, ColSize))
Else
    Set Rng_RsLev_ColNames = Range(Cells(1, 1), Cells(1, ColSize))
End If
Call z_CopyRange("All", "SmC_MasterDataSet_ResourceLevel", Rng_RsLev_ColNames,
"QualityChecksRS_FoundErrors", Sheets("QualityChecksRS_FoundErrors").Cells(1, 2))
End If
If flag Then
Call z_ExcelSessionWindowMinimized(wb_new)
Call z_ExcelSessionWindowNormal(wb_new)
MsgBox ("Look at the MasterPlan, check on/off and remove the filters!!")
Stop
Dim Array_ApplyBR() As Variant
Sheets("Master Plan").Activate
Dim RowSize_MasterPlan As Long
Dim col_BRStatus As Long
Dim col_BRNo As Long
col_BRStatus = z_GetColumnIndex("Business Rule Status", 4, "Master Plan")
col_BRNo = z_GetColumnIndex("RuleNo.", 4, "Master Plan")
RowSize_MasterPlan = z_RowSize(col_BRNo, "Master Plan")
ReDim Array_ApplyBR(0 To RowSize_MasterPlan - 4, 0 To 1) As Variant
For Row_MasterPlan = 5 To RowSize_MasterPlan
    Array_ApplyBR(Row_MasterPlan - 5, 0) = Cells(Row_MasterPlan, col_BRNo).Value
    Array_ApplyBR(Row_MasterPlan - 5, 1) = Cells(Row_MasterPlan, col_BRStatus).Value
Next Row_MasterPlan
End If

```

```
Call BusinessRules_Bundle1(Rng_PiLev, Rng_TkLev, Rng_RsLev, RowSize_MasterPlan,
Array_ApplyBR, wbdate, FirstYr)
```

```
Call BusinessRules_Bundle2(Rng_PiLev, Rng_TkLev, Rng_RsLev, RowSize_MasterPlan,
Array_ApplyBR, wbdate, FirstYr)
```

```
Call z_WorkbookSave(wb_new)
```

```
wb_new.Close
```

```
Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
Array("Main_CheckBusinessRules", "End", "task:", "file:", CStr(Now()), ""))
```

```
End Sub
```

```
Function BusinessRules_Bundle1(Rng_PiLev As Range, Rng_TkLev As Range, Rng_RsLev As Range,
RowSize_MasterPlan As Long, _
```

```
Array_ApplyBR As Variant, wbdate As String, FirstYr As Integer)
```

```
*****
```

```
'L&G
```

```
*****
```

```
Dim BusinessRuleNr As Integer
```

```
For Row_MasterPlan = 5 To RowSize_MasterPlan
```

```
'PI Lead AI
```

```
If Array_ApplyBR(Row_MasterPlan - 5, 0) = 1 And _
```

```
Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
```

```
BusinessRuleNr = 1
```

```
Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
```

```
"Portfolio Level 2", "LAWN_GARDEN", _
```

```
"Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
```

```
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
```

```
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
```

```
"PI Status", Array("Active", "Evaluation", "Planned"), _
```

```
"PI Sub Type", Array("AI & Product Maintenance", "AI New", "AI Re-registration",
```

```
"Formulation Extension", "Formulation New", "Label Extension") _
```

```
)
```

```
'Stop
```

```
Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
```

```
"QualityChecksPI_FoundErrors", _
```

```
BusinessRuleNr, "PI Lead AI", Array("="), Array("(blank)"))
```

```
'Stop
```

```
'PI List of Active Ingredients
```

```
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 2 And _
```

```
Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
```

```
BusinessRuleNr = 2
```

```
Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
```

```
"Portfolio Level 2", "LAWN_GARDEN", _
```

```
"Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
```

```
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
```

```
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
```

```
"PI Status", Array("Active", "Evaluation", "Planned"), _
```

```
"PI Sub Type", Array("AI & Product Maintenance", "AI New", "AI Re-registration",
```

```
"Formulation Extension", "Formulation New", "Label Extension") _
```

```
)
```

```
'Stop
```

```

        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI List of Active Ingredients", Array("="), "<>"), Array("(blank)", "Attribute:PI
Lead AI"))
    'Stop
    'BC Business Case Author
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 3 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 3
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active", "Evaluation", "Planned"), _
            "PI Sub Type", Array("Formulation New", "Label Extension", "Formulation Extension", "Idea
Evaluation", "Non-Product Customer Offer"), _
            ' , , , , , _
            "PI Syngenta Program", "<>PER" _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "BC Business Case Author", Array("="), Array("(blank)"))
    'Stop
    '?Terminal Value?
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 4 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 4
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active", "Planned"), _
            "PI Sub Type", Array("Formulation New", "Label Extension", "Formulation Extension", "Non-
Product Customer Offer"), _
            ' , , , , , _
            "PI Syngenta Program", "<>PER" _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "BC Terminal Value NPV (10% decline)", Array("="), Array("0"))
    'Stop
    'PI Customer Need
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 5 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 5
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _

```

```

        "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
        "PI Status", Array("Active", "Evaluation", "Planned"), _
        "PI Type", Array("Project") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Customer Need", Array("="), Array("(blank)"))
    'Stop
    'PI Responsibility
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 6 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 6
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active", "Evaluation", "Planned") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Responsibility", Array("=", "="), Array("NONE SELECTED", "CPD
FINANCE"))
    'Stop
    'PI Scope
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 7 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 7
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active", "Evaluation", "Planned") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Scope", Array("=", "<"), Array("(blank)", "NofCharacters:10"))
    'Stop
    'PI Syngenta Program
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 8 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 8
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _

```



```

        "PI Status", Array("Active", "Evaluation", "Planned") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Syngenta Program", Array("="), Array("(blank)"))
    'Stop
    'PI Customer
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 9 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 9
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active", "Evaluation", "Planned"), _
            "PI Sub Type", Array("Idea Evaluation") _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
            BusinessRuleNr, "PI Customer", Array("<>", "<>"), Array("MARKETING & SALES", "LAWN &
GARDEN"))
        'Stop
        'PI Customer
        Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 10 And _
            Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
            BusinessRuleNr = 10
            Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
                "Portfolio Level 2", "LAWN_GARDEN", _
                "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
                "PI Status", Array("Active", "Evaluation", "Planned"), _
                "PI Sub Type", Array("Formulation New", "Label Extension", "Formulation Extension", "AI
New") _
            )
            'Stop
            Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
                BusinessRuleNr, "PI Customer", Array("<>"), Array("LAWN & GARDEN"))
            'Stop
            'PI Customer
            Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 11 And _
                Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
                BusinessRuleNr = 11
                Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
                    "Portfolio Level 2", "LAWN_GARDEN", _
                    "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
                    "PI Status", Array("Active", "Evaluation", "Planned"), _

```

```

        "PI Sub Type", Array("AI & Product Maintenance", "AI Re-registration") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Customer", Array("<>"), Array("MARKETING & SALES"))
    'Stop
    'PI Customer
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 12 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 12
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active", "Evaluation", "Planned") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Customer", Array("="), Array("(blank)"))
    'Stop
    '-----
    'Activity Planned Start
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 13 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 13
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active"), _
            "TK Task Status", Array("Planned") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "Activity Planned Start", Array("<"), Array("Date:" &
z_Date_AsDateDoubleString(z_DateString_AsDate(wbdate) - 7)))
    'Stop
    'Activity Planned Start
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 14 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 14
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Evaluation"), _
            "TK Task Status", Array("Planned") _

```

```

    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "Activity Planned Start", Array("<"), Array("Date:" &
z_Date_AsDateDoubleString(z_DateString_AsDate(wbdate) - 7)))
    'Stop
    'Activity Expected finish
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 15 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 15
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active"), _
            "TK Task Status", Array("Active") _
            , , , , , , _
            "EAC Full Costs_c", ">0" _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "Activity Expected finish", Array("<"), Array("Date:" &
z_Date_AsDateDoubleString(z_DateString_AsDate(wbdate) - 7)))
    'Stop
    'TK Task Title
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 16 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 16
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active", "Evaluation", "Planned"), _
            "TK Task Status", Array("Active", "Planned") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Title", Array("="), Array("(blank)"))
    'Stop
    'TK Task Status
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 17 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 17
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _

```

```

        "PI Status", Array("Active", "Evaluation", "Planned") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Status", Array("="), Array("(blank)"))
    'Stop
    'TK Task Status
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 18 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 18
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Completed", "Terminated") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Status", Array("="), Array("Active"))
    'Stop
    'TK Task Status
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 19 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 19
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Completed", "Terminated") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Status", Array("="), Array("Planned"))
    'Stop
    'TK Task Status
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 20 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 20
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Evaluation") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _

```

```

        BusinessRuleNr, "TK Task Status", Array("=", "="), Array("Active", "Completed"))
    'Stop
'Sum of EAC Full Costs 2013_c - 2015_c
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 21 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 21
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "LAWN_GARDEN", _
        "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
        "PI Status", Array("Active", "Evaluation", "Planned", "Completed", "Terminated"), _
        "TK Task Status", Array("Completed", "Terminated") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForSeveralAttributes("SmC_FullDataSet_TkLevel",
Rng_TkLev, "QualityChecksTK_FoundErrors", _
        BusinessRuleNr, Array("EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 4) & "_c"), _
        Array(">", ">", ">"), Array("0", "0", "0"))
    'Stop
'TK Customer
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 22 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 22
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "LAWN_GARDEN", _
        "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
        "PI Status", Array("Active", "Evaluation", "Planned"), _
        "TK Task Status", Array("Terminated"), _
        ' , , , , , -
        "EAC Full Costs_c", ">0" _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Customer", Array("<>", "<>", "<>", "<>"), Array("MARKETING &
SALES", "CP DEVELOPMENT", "L&G - PLANT HEALTH", "L&G - FLOWERS")) "'Attribute:TK Customer"
    'Stop
'TK Customer
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 23 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 23
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "LAWN_GARDEN", _
        "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
        "PI Status", Array("Active", "Evaluation", "Planned"), _
        "TK Task Status", Array("Active", "Planned", "Completed") _

```

```

    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Customer", Array("<>", "<>", "<>", "<>"), Array("MARKETING &
SALES", "CP DEVELOPMENT", "L&G - PLANT HEALTH", "L&G - FLOWERS"))
    'Stop
    'Activity Actual Start
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 24 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 24
        Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active"), _
            "TK Task Status", Array("Active"), _
            "RBS Level 1", Array("Product Safety"), _
            ' , , , , _
            "Activity Planned Start", "<" & z_Date_AsDateDoubleString(z_DateString_AsDate(wbdate) -
7) _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, "Activity Actual Start", Array("="), Array("(blank)"))
    'Stop
    'TK Task Contact
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 25 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 25
        Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
            "Portfolio Level 2", "LAWN_GARDEN", _
            "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
            "PI Status", Array("Active"), _
            "TK Task Status", Array("Active"), _
            "RBS Level 1", Array("Product Safety"), _
            ' , , , , _
            "Activity Planned Start", "<" & z_Date_AsDateDoubleString(z_DateString_AsDate(wbdate) -
7) _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, "TK Task Contact", Array("="), Array("(blank)"))
    'Stop
    'TK Task Location
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 26 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 26

```

```

Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
    "Portfolio Level 2", "LAWN_GARDEN", _
    "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
    "PI Status", Array("Active"), _
    "TK Task Status", Array("Active"), _
    "RBS Level 1", Array("Product Safety"), _
    '','','_
    "Activity Planned Start", "<" & z_Date_AsDateDoubleString(z_DateString_AsDate(wbdate) -
7) _
    )
'Stop
Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
    BusinessRuleNr, "TK Task Location", Array("="), Array("(blank)"))
'Stop
*****

'CPD
*****

'PI Lead AI
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 27 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 27
    Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active", "Planned"), _
        "PI Sub Type", Array("Formulation New", "Label Extension", "Formulation Extension", "AI &
Product Maintenance", "AI New", "AI Re-registration") _
    )
'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Lead AI", Array("=", "="), Array("(blank)", "Not Applicable"))
'Stop
'PI List of Active Ingredients
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 28 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 28
    Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active", "Planned"), _
        "PI Sub Type", Array("Formulation New", "Label Extension", "Formulation Extension", "AI &
Product Maintenance", "AI New", "AI Re-registration") _
    )
'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI List of Active Ingredients", Array("=", "<>"), Array("(blank)", "Attribute:PI
Lead AI"))
'Stop
'BC Business Case Author
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 29 And _

```

```

        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
BusinessRuleNr = 29
Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
    "Portfolio Level 2", "CPD", _
    "PI Status", Array("Active", "Planned"), _
    "PI Sub Type", Array("Formulation New", "Label Extension", "Formulation Extension", "Idea
Evaluation", "Non-Product Customer Offer"), _
    , , , , , , , _
    "PI Syngenta Program", "<>PER" _
)
'Stop
Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
    BusinessRuleNr, "BC Business Case Author", Array("="), Array("(blank)"))
'Stop
'?Terminal Value?
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 30 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
BusinessRuleNr = 30
Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
    "Portfolio Level 2", "CPD", _
    "PI Status", Array("Active", "Planned"), _
    "PI Sub Type", Array("Formulation New", "Label Extension", "Formulation Extension", "Non-
Product Customer Offer"), _
    , , , , , , , _
    "PI Syngenta Program", "<>PER" _
)
'Stop
Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
    BusinessRuleNr, "BC Terminal Value NPV (10% decline)", Array("="), Array("0"))
'Stop
'PI Customer Need
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 31 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
BusinessRuleNr = 31
Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
    "Portfolio Level 2", "CPD", _
    "PI Status", Array("Active", "Planned"), _
    "PI Type", Array("Project") _
)
'Stop
Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
    BusinessRuleNr, "PI Customer Need", Array("="), Array("(blank)"))
'Stop
'PI Responsibility
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 32 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
BusinessRuleNr = 32
Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
    "Portfolio Level 2", "CPD", _
    "PI Status", Array("Active", "Planned") _

```



```

    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Responsibility", Array("=", "="), Array("NONE SELECTED", "CPD
FINANCE"))
    'Stop
    'PI Scope
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 33 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 33
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active", "Planned") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Scope", Array("=", "<"), Array("(blank)", "NofCharacters:10"))
    'Stop
    'PI Syngenta Program
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 34 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 34
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active", "Planned") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Syngenta Program", Array("=", ""), Array("(blank)"))
    'Stop
    'PI Customer
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 35 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 35
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active", "Planned") _
        )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Customer", Array("<>", "<>"), Array("MARKETING & SALES", "CP
DEVELOPMENT"))
    'Stop
    '?PI list of Countries?
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 36 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 36
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "CPD", _

```

```

        "PI Status", Array("Active", "Planned") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI List of Countries (PI Geography)", Array("="), Array("(blank)"))
    'Stop
    'PI list of Crops
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 37 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 37
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active", "Planned"), _
            "PI Sub Type", Array("Formulation New", "Label Extension", "AI New", "Formulation
Extension", "AI & Product Maintenance", "Idea Evaluation") _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
            BusinessRuleNr, "PI list of Crops", Array("="), Array("(blank)"))
        'Stop
        '-----

    'TK Customer
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 38 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 38
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active", "Planned"), _
            "TK Task Status", Array("Active", "Planned", "Completed", "Terminated", "(blank)", _
            ' , , , , , , , ' _
            "EAC Full Costs Current Year & Future Years", "=0.00" _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
            BusinessRuleNr, "TK Task Customer", Array("="), Array("(blank)"))
        'Stop
        'Activity Type
        Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 39 And _
            Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
            BusinessRuleNr = 39
            Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
                "Portfolio Level 2", "CPD", _
                "PI Status", Array("Active", "Planned"), _
                "TK Task Status", Array("Active", "Planned") _
            )
            'Stop
            Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
                BusinessRuleNr, "Activity Type", Array("="), Array("(blank)"))

```

```

'Stop
'TK Task Status
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 40 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 40
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active", "Planned") _
    )
'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Status", Array("="), Array("(blank)"))
'Stop
'TK Task Title
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 41 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 41
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active", "Planned"), _
        "TK Task Status", Array("Active", "Planned") _
    )
'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Title", Array("="), Array("(blank)"))
'Stop
'TK Task Contact
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 42 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 42
    Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active"), _
        "TK Task Status", Array("Active"), _
        "RBS Level 1", Array("Product Safety"), _
        '','',''_
        "Activity Actual Start", "<>(blank)" _
    )
'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, "TK Task Contact", Array("="), Array("(blank)"))
'Stop
'TK Task Location
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 43 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 43
    Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active"), _
        "TK Task Status", Array("Active"), _

```

```

        "RBS Level 1", Array("Product Safety"), _
        , , , , , _
        "Activity Actual Start", "<>(blank)" _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, "TK Task Location", Array("="), Array("(blank)"))
    'Stop
    ,

Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 44 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 44
    Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active"), _
        "TK Task Status", Array("Active"), _
        "RBS Level 1", Array("Product Safety"), _
        , , , , , _
        "Activity Actual Start", "<>(blank)" _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, "EAC Full Costs Current Year & Future Years", Array("="), Array("0"), _
        "SmC_FullDataSet_TkLevel", Rng_TkLev, "TkIdentifier", "EAC Full Costs Current Year &
Future Years")
    'Stop
    ,

Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 45 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 45
    'z_AddAuxiliaryAttribute_Sum(.."EAC Full Costs Current Year & Future Years"..) calculated in 38
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active", "Planned", "Completed", "Terminated"), _
        "TK Task Status", Array("Active", "Planned", "Completed", "Terminated", "(blank)", _
        , , , , , , , _
        "EAC Full Costs Current Year & Future Years", ">0.00" _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Customer", Array("="), Array("(blank)"))
    'Stop
    ,

Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 46 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 46
    'z_AddAuxiliaryAttribute_Sum(.."EAC Full Costs Current Year & Future Years"..) calculated in 38
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Completed", "Terminated"), _

```

```

        , , , , , , , , , , _
        "EAC Full Costs Current Year & Future Years", ">0.00" _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Status", Array("="), Array("(blank)"))
    'Stop
,

ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 47 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 47
    Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active", "Planned", "Completed", "Terminated"), _
        "TK Task Status", Array("Completed", "Terminated"), _
        "RBS Level 1", Array("Product Safety"), _
        , , , , , , , , , , _
        "Activity Actual Finish", "<>(blank)" _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForSeveralAttributes("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, Array("EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 4) & "_c"), _
        Array(">", ">", ">"), Array("0", "0", "0"))
    'Stop
,

ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 48 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 48
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Completed", "Terminated") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Status", Array("="), Array("Active"))
    'Stop
End If
Next Row_MasterPlan

```

End Function

```

Function BusinessRules_Bundle2(Rng_PiLev As Range, Rng_TkLev As Range, Rng_RsLev As Range,
RowSize_MasterPlan As Long, _
    Array_ApplyBR As Variant, wbddate As String, FirstYr As Integer)
    *****
    'L&G
    *****

```

Dim BusinessRuleNr As Integer

For Row\_MasterPlan = 5 To RowSize\_MasterPlan

```
*****
'CPR
*****
,
If Array_ApplyBR(Row_MasterPlan - 5, 0) = 49 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 49
    Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
        "Portfolio Level 2", "CPR", _
        "PI Status", Array("Active", "Planned") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Scope", Array("=", "<"), Array("(blank)", "NofCharacters:10"))
    'Stop
,
ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 50 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 50
    Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
        "Portfolio Level 2", "CPR", _
        "PI Status", Array("Active", "Planned") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Syngenta Program", Array("="), Array("(blank)"))
    'Stop
,
ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 51 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 51
    Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
        "Portfolio Level 2", "CPR", _
        "PI Status", Array("Active", "Planned") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Customer", Array("<>"), Array("CP RESEARCH"))
    'Stop
,
ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 52 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 52
    Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
        "Portfolio Level 2", Array("CPR", "CPD", "LAWN_GARDEN"), _
        "PI Status", Array("Active", "Planned") _
```

```

    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
        BusinessRuleNr, "PI Lead Strategic Crop", Array("="), Array("(blank)"))
    'Stop
    ,
    Elself Array_ApplyBR(Row_MasterPlan - 5, 0) = 53 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 53
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", Array("CPR", "CPD", "LAWN_GARDEN"), _
            "PI Status", Array("Active", "Planned") _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
            BusinessRuleNr, "PI Strategic Crop Coefficient", Array("<>"), Array("1"))
        'Stop
        ,
        Elself Array_ApplyBR(Row_MasterPlan - 5, 0) = 54 And _
            Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
            BusinessRuleNr = 54
            Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
                "Portfolio Level 2", "CPR", _
                "PI Status", Array("Active", "Planned", "Completed", "Terminated", "Evaluation"), _
                "PI Last Gate Passed", Array("G1.22 - LATE LEAD EXPLORATION", "G1.3 - OPTIMIZATION") _
            )
            'Stop
            Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
                BusinessRuleNr, "PI URL", Array("="), Array("(blank)"))
            'Stop
            '-----
            ,
            Elself Array_ApplyBR(Row_MasterPlan - 5, 0) = 55 And _
                Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
                BusinessRuleNr = 55
                'z_AddAuxiliaryAttribute_Sum(.."EAC Full Costs Current Year & Future Years"..) calculated in 38
                Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
                    "Portfolio Level 2", "CPR", _
                    "PI Status", Array("Active", "Planned", "Evaluation", "Completed", "Terminated"), _
                    "TK Task Status", Array("Active", "Planned", "Completed", "Terminated", "(blank)"), _
                    , , , , , , _
                    "EAC Full Costs Current Year & Future Years", ">0.00" _
                )
                'Stop
                Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
                    BusinessRuleNr, "TK Task Customer", Array("="), Array("(blank)"))
                'Stop
                ,
                Elself Array_ApplyBR(Row_MasterPlan - 5, 0) = 56 And _

```

```

        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 56
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "CPR", _
            "PI Status", Array("Active", "Planned"), _
            "TK Task Status", Array("Active", "Planned")) _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
            BusinessRuleNr, "Activity Type", Array("="), Array("(blank)"))
        'Stop
    ,

    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 57 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 57
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "CPR", _
            "PI Status", Array("Active", "Planned")) _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
            BusinessRuleNr, "TK Task Status", Array("="), Array("(blank)"))
        'Stop
    ,

    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 58 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 58
        'z_AddAuxiliaryAttribute_Sum(.."EAC Full Costs Current Year & Future Years"..) calculated in 38
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "CPR", _
            "PI Status", Array("Active", "Planned"), _
            "TK Task Status", Array("Active", "Planned"), _
            ' , , , , , , , , _
            "EAC Full Costs Current Year & Future Years", ">0.00" _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
            BusinessRuleNr, "TK Task Status", Array("="), Array("(blank)"))
        'Stop
    ,

    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 59 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 59
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "CPR", _
            "PI Status", Array("Active", "Planned"), _
            "TK Task Status", Array("Active", "Planned")) _
        )
        'Stop

```



```

        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Title", Array("="), Array("(blank)"))
    'Stop
,
ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 60 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 60
    Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
        "Portfolio Level 2", "CPR", _
        "PI Status", Array("Active"), _
        "TK Task Status", Array("Active"), _
        "RBS Level 1", Array("Product Safety") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, "TK Task Contact", Array("="), Array("(blank)"))
    'Stop
,
ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 61 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 61
    Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
        "Portfolio Level 2", "CPR", _
        "PI Status", Array("Active"), _
        "TK Task Status", Array("Active"), _
        "RBS Level 1", Array("Global Supply Technology & Projects") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, "TK Task Location", Array("="), Array("(blank)"))
    'Stop
,
ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 62 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 62
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "CPR", _
        "PI Status", Array("Active"), _
        "TK Task Status", Array("Active") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "EAC Full Costs Current Year & Future Years", Array("="), Array("0"))
    'Stop
,
ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 63 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 63
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _

```

```

        "Portfolio Level 2", "CPR", _
        "PI Status", Array("Active", "Planned", "Evaluation", "Completed", "Terminated", "(blank)"),
    -
        "TK Task Status", Array("Completed", "Terminated") _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForSeveralAttributes("SmC_FullDataSet_TkLevel",
Rng_TkLev, "QualityChecksTK_FoundErrors", _
        BusinessRuleNr, Array("EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 4) & "_c"), _
        Array(">", ">", ">"), Array("0", "0", "0"))
    'Stop
    ,
    ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 64 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 64
        Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active", "Planned", "Completed", "Terminated"), _
            "TK Task Status", Array("Completed", "Terminated") _
            , , , , , _
            "Activity Actual Start", "(blank)", , , _
            "Activity Actual Finish", "(blank)" _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForSeveralAttributes("SmC_FullDataSet_TkLevel",
Rng_TkLev, "QualityChecksTK_FoundErrors", _
            BusinessRuleNr, Array("EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
                "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
                "EAC Full Costs " & CStr(FirstYr + 4) & "_c"), _
            Array(">", ">", ">"), Array("0", "0", "0"))
        'Stop
    ,
    ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 65 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 65
        Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active", "Planned", "Completed", "Terminated"), _
            "TK Task Status", Array("Completed", "Terminated"), _
            "RBS Level 1", Array("Product Safety"), _
            , , , , , _
            "Activity Actual Start", "<>(blank)" _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForSeveralAttributes("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
            BusinessRuleNr, Array("EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
                "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
                "EAC Full Costs " & CStr(FirstYr + 4) & "_c"), _
            Array(">", ">", ">"), Array("0", "0", "0"))
        'Stop

```

```

,
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 66 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 66
    'z_AddAuxiliaryAttribute_Sum(.."EAC Full Costs Current Year & Future Years"..) calculated in 38
    Call z_SetFilter("SmC_FullDataSet_TkLevel", Rng_TkLev, _
        "Portfolio Level 2", "LAWN_GARDEN", _
        "PI Status", Array("Completed", "Terminated"), _
        , , , , , , , , _
        "EAC Full Costs Current Year & Future Years", ">0.00" _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_TkLevel", Rng_TkLev,
"QualityChecksTK_FoundErrors", _
        BusinessRuleNr, "TK Task Status", Array("="), Array("(blank)"))
    'Stop
    'TK Task Contact
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 67 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 67
        Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active"), _
            "TK Task Status", Array("Active"), _
            "RBS Level 1", Array("Global Supply Technology & Projects"), _
            , , , , , , _
            "Activity Actual Start", "<>(blank)" _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
            BusinessRuleNr, "TK Task Contact", Array("="), Array("(blank)"))
        'Stop

    'TK Task Location
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 68 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 68
        Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active"), _
            "TK Task Status", Array("Active"), _
            "RBS Level 1", Array("Global Supply Technology & Projects"), _
            , , , , , , _
            "Activity Actual Start", "<>(blank)" _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
            BusinessRuleNr, "TK Task Location", Array("="), Array("(blank)"))
        'Stop
    ,
    Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 69 And _

```

```

        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
BusinessRuleNr = 69
Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
    "Portfolio Level 2", "CPD", _
    "PI Status", Array("Active"), _
    "TK Task Status", Array("Active"), _
    "RBS Level 1", Array("Global Supply Technology & Projects"), _
    , , , , , _
    "Activity Actual Start", "<>(blank)" _
)
'Stop
Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
    BusinessRuleNr, "EAC Full Costs Current Year & Future Years", Array("="), Array("0"), _
    "SmC_FullDataSet_TkLevel", Rng_TkLev, "TkIdentifier", "EAC Full Costs Current Year &
Future Years")
'Stop
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 70 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
BusinessRuleNr = 70
Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
    "Portfolio Level 2", "CPD", _
    "PI Status", Array("Active"), _
    "TK Task Status", Array("Active"), _
    "RBS Level 1", Array("Global Supply Technology & Projects", "Product Safety"), _
    , , , , , _
    "Activity Actual Start", "(blank)" _
)
'Stop
Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
    BusinessRuleNr, "EAC Full Costs Current Year & Future Years", Array("="), Array("0"), _
    "SmC_FullDataSet_TkLevel", Rng_TkLev, "TkIdentifier", "EAC Full Costs Current Year &
Future Years")
'Stop
,
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 71 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
BusinessRuleNr = 71
Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
    "Portfolio Level 2", "CPD", _
    "PI Status", Array("Active"), _
    "TK Task Status", Array("Active"), _
    , , , , , , , _
    "RBS Level 1", "<>Global Supply Technology & Projects", "xlAnd", "<>Product Safety" _
)
'Stop
Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
    BusinessRuleNr, "EAC Full Costs Current Year & Future Years", Array("="), Array("0"), _
    "SmC_FullDataSet_TkLevel", Rng_TkLev, "TkIdentifier", "EAC Full Costs Current Year &
Future Years")
'Stop

```

```

'
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 72 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 72
    Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active", "Planned", "Completed", "Terminated"), _
        "TK Task Status", Array("Completed", "Terminated"), _
        "RBS Level 1", Array("Global Supply Technology & Projects"), _
        "Activity Actual Start", "(blank)", _
        ', , , , _
        "Activity Actual Finish", "<>(blank)" _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForSeveralAttributes("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, Array("EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 4) & "_c"), _
            Array(">", ">", ">"), Array("0", "0", "0"))
    'Stop
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 73 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 73
    Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
        "Portfolio Level 2", "CPD", _
        "PI Status", Array("Active", "Planned", "Completed", "Terminated"), _
        "TK Task Status", Array("Completed", "Terminated"), _
        "RBS Level 1", Array("Global Supply Technology & Projects"), _
        ', , , , , _
        "Activity Actual Start", "<>(blank)" _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForSeveralAttributes("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, Array("EAC Full Costs " & CStr(FirstYr + 2) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
            "EAC Full Costs " & CStr(FirstYr + 4) & "_c"), _
            Array(">", ">", ">"), Array("0", "0", "0"))
    'Stop
    'Activity Actual Start
Elseif Array_ApplyBR(Row_MasterPlan - 5, 0) = 74 And _
    Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
    BusinessRuleNr = 74
    Call z_SetFilter("SmC_MasterDataSet_ResourceLevel", Rng_RsLev, _
        "Portfolio Level 2", "LAWN_GARDEN", _
        "Portfolio Level 3", Array("GARDEN_CONTROL", "HOME_PEST_CONTROL",
"MATERIALS_PROTECTION", "ORNAMENTALS", "PLANT_HEALTH", "PROF_GROW_MEDIA",
"PROF_PEST_MNGT", "TURF", "VECTOR_CONTROL", "VEGETATION_MANAGEMENT"), _
        "PI Status", Array("Active"), _
        "TK Task Status", Array("Active"), _
        "RBS Level 1", Array("Global Supply Technology & Projects"), _
        ', , , , _

```

```

        "Activity Planned Start", "<" & z_Date_AsDateDoubleString(z_DateString_AsDate(wbdate) -
7) _
    )
    'Stop
    Call z_QualityCheckBusinessRule_ForAttribute("SmC_MasterDataSet_ResourceLevel",
Rng_RsLev, "QualityChecksRS_FoundErrors", _
        BusinessRuleNr, "Activity Actual Start", Array("="), Array("(blank)"))
    'Stop
    'PI Lead AI
    ElseIf Array_ApplyBR(Row_MasterPlan - 5, 0) = 75 And _
        Array_ApplyBR(Row_MasterPlan - 5, 1) = "On" Then
        BusinessRuleNr = 75
        Call z_SetFilter("SmC_FullDataSet_PiLevel", Rng_PiLev, _
            "Portfolio Level 2", "CPD", _
            "PI Status", Array("Active", "Planned"), _
            "PI Sub Type", Array("Unplanned Resource", "Non-Product Customer Offer", "Capability &
Technology Development", "Pack Development", "Idea Evaluation", "Stewardship") _
        )
        'Stop
        Call z_QualityCheckBusinessRule_ForAttribute("SmC_FullDataSet_PiLevel", Rng_PiLev,
"QualityChecksPI_FoundErrors", _
            BusinessRuleNr, "PI Lead AI", Array("="), Array("(blank)"))
        'Stop
    End If
Next Row_MasterPlan

```

End Function

```

Sub StartAll_QualCheckGenerateInput()
    Dim wbdate As String
    wbdate = "2012_10_16"
    Call m_Main_QualCheckGenerateInput(wbdate)
End Sub

```

```

Sub m_Main_QualCheckGenerateInput(Optional wbdate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
    *****

    Dim flag As Integer
    'Dim wbdate As String
    Dim reportversion_folder As String
    Dim reportversion_file As String
    *****

    'reopen
    flag = 1
    flag_OpenExistingWbs = 1
    If wbdate = Empty Then
        wbdate = "2012_10_16" & VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) & "2012_2_8"
    End If
    reportversion_folder = "_V1-0"
    reportversion_file = "_V1-0"

```

```

folderdescription = ""
*****

'Open and activate an Excel workbook (and session)
Dim WbPath As String
Dim WbName As String
Dim wb_new As Workbook
Dim Wb_QualityChecksBR As Workbook
Dim Wb_MasterPlan As Workbook
*****
*****

'Adds the new workbook RD_MasterDataSet
*****
*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_QualityCheckInput_" & wdate &
reportversion_file & ".xlsb" 'for test purposes
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
End If

'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_QualityCheckBusinessRules_" &
wdate & reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_QualityChecksBR)
End If

'copy a sheet from another workbook
If flag_OpenExistingWbs Then
    Call z_CopySheetFromWb1ToWb2("Master Plan", Wb_QualityChecksBR, wb_new, "Begin")
    Call z_CopySheetFromWb1ToWb2("QualityChecksPI_FoundErrors", Wb_QualityChecksBR,
wb_new, "Begin")
    Call z_CopySheetFromWb1ToWb2("QualityChecksTK_FoundErrors", Wb_QualityChecksBR,
wb_new, "Begin")
    Call z_CopySheetFromWb1ToWb2("QualityChecksRS_FoundErrors", Wb_QualityChecksBR,
wb_new, "Begin")
    Wb_QualityChecksBR.Close False
End If

If flag Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", wb_new)
        Call z_ShDelete("Sheet2", wb_new)
        Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If

'flat value copy of sheets (entries will be deleted in the function
z_Main_QualityCheckSummarySorted)
If flag_OpenExistingWbs Then

```

```

    Call z_ShNewFlatValueCopy("QualityChecksPI_FoundErrors", "QualityCheckPiRsSorted", "Begin")
    Call z_ShNewFlatValueCopy("QualityChecksTK_FoundErrors", "QualityCheckTkRsSorted", "Begin")
End If
*****

If flag Then
    'generate the sheets with the errors as one line per Id
    'delete rows with same Id
    'delete columns without errors, but do not delete those in the exception arrays (defined within the
function, read from the wb AttributeNames_FromSheet_ToArray.xlsb)
    Call z_Main_QualityCheckSummarySorted("QualityChecksPI_FoundErrors",
"QualityChecksTK_FoundErrors", "QualityChecksRS_FoundErrors", _
    "QualityCheckPiRsSorted", "QualityCheckTkRsSorted")
End If
*****

If flag Then
    'update ActionComments in Master Plan
    Call z_UpdateActionComments("Master Plan", 4, "Action for Recipient", "RuleNo.", False)
End If
If flag Then
    'Add ActionComments to QualityCheck files
    Dim Row_to As Long
    Call z_ShAdd_ActionCommentsAndAdditionalAttributes("Master Plan", 4, _
        "RuleNo.", "Attribute (DataSet)", "Action for Recipient", _
        "Recipient", "Deputy Recipient", "Additional Columns", _
        "QualityCheckPiRsSorted", "BusinessRuleNr")
    Call z_ShAdd_ActionCommentsAndAdditionalAttributes("Master Plan", 4, _
        "RuleNo.", "Attribute (DataSet)", "Action for Recipient", _
        "Recipient", "Deputy Recipient", "Additional Columns", _
        "QualityCheckTkRsSorted", "BusinessRuleNr")
End If
*****

    Call z_WorkbookSave(wb_new)
    wb_new.Close True
End Sub

```

```

Sub m_StartAll_Main_GenerateTheRecipientAttachement()
    'pw for code: dqcode
    Dim wbdate As String
    wbdate = "2012_10_16"
    Call m_Main_GenerateTheRecipientAttachement(wbdate)
End Sub

```

```

Sub m_Main_GenerateTheRecipientAttachement(Optional wbdate As String)
    Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
    *****

    Dim flag As Integer
    'Dim wbdate As String

```



```

Dim reportversion_folder As String
Dim reportversion_file As String
*****

'reopen
flag = 1
flag_OpenExistingWbs = 1
If wbdate = Empty Then
    wbdate = "2012_10_16" & VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) & "2012_2_8"
End If
reportversion_folder = "_V1-0"
reportversion_file = "_V1-0"
folderdescription = ""
*****

'Open and activate an Excel workbook (and session)
Dim WbPath As String
Dim WbName As String
Dim wb_new As Workbook
Dim Wb_QualityCheckInput As Workbook
Dim Wb_DataQualityHelp As Workbook
*****
*****
*****

'Adds the new workbook RD_MasterDataSet
*****
*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_DataQualityTool_" & wbdate &
reportversion_file & ".xlsb" 'for test purposes
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
End If
'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_QualityCheckInput_" & wbdate &
reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_QualityCheckInput)
End If
'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
    WbName = "DataQualityHelp.xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_DataQualityHelp)
'seems to help to protect an error
    Call z_WorkbookSave(Wb_DataQualityHelp)
End If
'copy a sheet from another workbook
If flag_OpenExistingWbs Then

```

```

    Call z_CopySheetFromWb1ToWb2("QualityCheckPiRsSorted", Wb_QualityCheckInput, wb_new,
"Begin")
    Call z_CopySheetFromWb1ToWb2("QualityCheckTkRsSorted", Wb_QualityCheckInput, wb_new,
"Begin")
    Call z_CopySheetFromWb1ToWb2("Master Plan", Wb_QualityCheckInput, wb_new, "Begin")
    Call z_CopySheetFromWb1ToWb2("Help", Wb_DataQualityHelp, wb_new, "Begin")
    'read the row no of found error lines
    Wb_QualityCheckInput.Activate
    Dim NofPiErrors As Long
    Dim NofTkErrors As Long
    Dim NofRsErrors As Long
    NofPiErrors = z_RowSize(1, "QualityChecksPI_FoundErrors") - 1
    NofTkErrors = z_RowSize(1, "QualityChecksTK_FoundErrors") - 1
    NofRsErrors = z_RowSize(1, "QualityChecksRS_FoundErrors") - 1
    wb_new.Activate
    'close Wb_GenPart1, closes the active workbook and saves any changes
    Wb_QualityCheckInput.Close False
    Wb_DataQualityHelp.Close False
End If

```

```

If flag_OpenExistingWbs Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", wb_new)
        Call z_ShDelete("Sheet2", wb_new)
        Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If

```

```

If flag Then
    Call z_ShNew("Output", "Begin", , , "Delete")
    Call z_ShNew("Input", "Begin", , , "Delete")
End If

```

'Generate sheet "Input" with dropdown list and linked button to macro

```

If flag Then
    Dim Text_ As String

    '***generate recipient list and list with all errors
    'create name list for dropdown list and error list and hide it
    Dim Rng_from As Range
    Sheets("Input").Columns(2).Hidden = False
    Sheets("Input").Columns("M:S").Hidden = False
    Call z_CreateListOfRecipients("Input", 2, 13, 14, 15, 16, 17, 18, 19, _
        "QualityCheckPiRsSorted", "QualityCheckTkRsSorted", _
        "PiIdentifier", "Recipient", "Deputy Recipient", "BusinessRuleNr", "Additional Columns", _
        "Master Plan", "RuleNo.", "Attribute (DataSet)")
    Dim RowSize As Long
    RowSize = z_RowSize(2, "Input")
    Set Rng_from = Sheets("Input").Range( _
        Sheets("Input").Cells(2, 2), _
        Sheets("Input").Cells(RowSize, 2))

```

```

***add some formatting and text
'row and column size, merge and interior color
Sheets("Input").Activate
Cells.Select
Selection.Interior.Color = RGB(255, 255, 255)
Selection.EntireColumn.ColumnWidth = 8.5
Cells(1, 1).Select
Cells(1, 3).EntireColumn.ColumnWidth = 4
Cells(1, 4).EntireColumn.ColumnWidth = 4
Cells(1, 7).EntireColumn.ColumnWidth = 17
Cells(1, 10).EntireColumn.ColumnWidth = 4
Cells(1, 11).EntireColumn.ColumnWidth = 4
Cells(3, 1).EntireRow.RowHeight = 30
Cells(4, 1).EntireRow.RowHeight = 30
Cells(21, 1).EntireRow.RowHeight = 80
Range(Cells(21, 5), Cells(21, 10)).Merge
Range(Cells(23, 5), Cells(23, 7)).Merge
Range(Cells(11, 5), Cells(12, 7)).Merge
Range(Cells(2, 3), Cells(24, 11)).BorderAround Weight:=xlThick, Color:=RGB(200, 200, 200)
Range(Cells(2, 3), Cells(7, 11)).Interior.Color = RGB(192, 80, 77)
Range(Cells(8, 3), Cells(24, 11)).Interior.Color = RGB(230, 230, 230)
Range(Cells(11, 5), Cells(12, 7)).Interior.Color = RGB(255, 255, 255)
'texts
Cells(3, 10).Value = "SmartChoice"
Cells(3, 10).Font.name = "Calibri"
Cells(3, 10).Font.Color = RGB(255, 255, 255)
Cells(3, 10).Font.Size = "26"
Cells(3, 10).Font.Bold = True
Cells(3, 10).HorizontalAlignment = xlRight
Cells(3, 10).VerticalAlignment = xlCenter
,

Cells(4, 10).Value = "Data Quality Report"
Cells(4, 10).Font.name = "Calibri"
Cells(4, 10).Font.Color = RGB(255, 255, 255)
Cells(4, 10).Font.Size = "26"
Cells(4, 10).Font.Bold = True
Cells(4, 10).HorizontalAlignment = xlRight
Cells(4, 10).VerticalAlignment = xlCenter
,

Cells(9, 5).Value = "Select your Name from the drop-down list"
Cells(9, 5).Font.name = "Arial"
Cells(9, 5).Font.Size = "12"
Cells(9, 5).HorizontalAlignment = xlLeft
Cells(9, 5).VerticalAlignment = xlCenter
,

Cells(14, 5).Value = "Click on the «Go» button"
Cells(14, 5).Font.name = "Arial"
Cells(14, 5).Font.Size = "12"
Cells(14, 5).Characters(14, 4).Font.Bold = True
Cells(14, 5).HorizontalAlignment = xlLeft
Cells(14, 5).VerticalAlignment = xlCenter
,

```

```

Cells(19, 5).Value = "(Please wait, this process may take up to 2 minutes)"
Cells(19, 5).Font.name = "Arial"
Cells(19, 5).Font.Size = "10"
Cells(19, 5).Font.Bold = False
Cells(19, 5).Font.Italic = True
Cells(19, 5).HorizontalAlignment = xlLeft
Cells(19, 5).VerticalAlignment = xlCenter
'

Cells(21, 5).Value = "The sheet «Output» shows all projects and tasks that may contain one or
more errors." & vbLf & vbLf & "Please go to SmartChoice and correct all indicated errors by
following the action in the comment box."
Cells(21, 5).Font.name = "Arial"
Cells(21, 5).Font.Size = "12"
Cells(21, 5).Characters(13, 8).Font.Bold = True
Cells(21, 5).HorizontalAlignment = xlLeft
Cells(21, 5).VerticalAlignment = xlTop
'

Range(Cells(23, 5), Cells(23, 7)).Select
ActiveSheet.Hyperlinks.Add Anchor:=Selection, Address:= _
"http://smartchoice.pro.intra/", TextToDisplay:= "SmartChoice Production"
Cells(23, 5).Font.name = "Arial"
Range(Cells(23, 5), Cells(23, 7)).Font.Size = "12"
Range(Cells(23, 5), Cells(23, 7)).HorizontalAlignment = xlLeft
Range(Cells(23, 5), Cells(23, 7)).VerticalAlignment = xlCenter
'import the picture
'Dim Pic As Picture
'Cells(3, 4).Select
'(will be inserted as link)
'Set Pic = Sheets("Input").Pictures.Insert( _
"C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\ZZZ_Inputs_MainGenerateReport\
LOGO Data Quality.png", True)
'Pic.height = 3.19 * 28.5
'Pic.width = 6.38 * 28.5

Dim imagePath As String
imagePath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\LOGO Data Quality.png"
Sheets("Input").Shapes.AddPicture Filename:=imagePath, _
LinkToFile:=False, _
SaveWithDocument:=True, _
left:=122, _
top:=30, _
width:=182, _
height:=91

'add bullets and arrow
Set shp = z_AddBulletShape("Input", "1", _
123, 150, 17, 17, msoShapeOval)
Set shp = z_AddBulletShape("Input", "2", _
123, 225, 17, 17, msoShapeOval)
Set shp = z_AddBulletShape("Input", "3", _
123, 330, 17, 17, msoShapeOval)
Set shp = z_AddArrowShape("Input", "", _

```

123, 424, 17, 17, msoShapeRightArrow)

```
***generate dropdown list and link it to the recipient list
'add dropdown list
Sheets("Input").Activate
Dim Cell_to As Range
Set Cell_to = Sheets("Input").Range(Cells(11, 5), Cells(12, 7))
Cell_to.Select
Call z_deleteDropDownList("Input", Cell_to)
Call z_addDropDownList("Input", Cell_to, Rng_from)
'add formatting
Sheets("Input").Activate
Cell_to.Font.name = "Arial"
Cell_to.Font.Bold = False
Cell_to.Font.Size = "14"
Cell_to.HorizontalAlignment = xlCenter
Cell_to.VerticalAlignment = xlCenter
'add input message
Text_ = "Dear User, Please click on the arrow and select your name from the dropdown list."
'Call z_deleteInputMessage("Input", Cell_to)
'Call z_addInputMessage("Input", Cell_to, Text_)
'zoom window to 120%
Sheets("Input").Activate
ActiveWindow.Zoom = 120

***generate macro with filtering code
'Stop
Call z_CreateModuleAndCode(wb_new)
'Stop
***generate button and link it to the macro
'add shape button with procedure onAction
Dim Btn As Shape
Text_ = "Go"
Set Btn = z_AddShapeAssignMacro("Input", Text_, "m_showRecipientsErrors", _
    148, 252, 70, 30, msoShapeRectangle)

'hide columns
Sheets("Input").Columns(2).Hidden = True
Sheets("Input").Columns("M:S").Hidden = True

***output template
'headers
Sheets("Output").Cells(2, 2) = "Recipient"
Sheets("Output").Cells(2, 3) = "PI Identifier"
Sheets("Output").Cells(2, 4) = "PI Title"
Sheets("Output").Cells(2, 5) = "PI Status"
Sheets("Output").Cells(2, 6) = "Task Identifier"
Sheets("Output").Cells(2, 7) = "Task Title"
Sheets("Output").Cells(2, 8) = "Task Status"
'formatting
Sheets("Output").Activate
Rows(1).RowHeight = 10
Columns(1).ColumnWidth = 2
```

```

Rows(2).RowHeight = 30
Columns("B:U").ColumnWidth = 12
Dim Rng_Header As Range
Dim Rng_Body As Range
Sheets("Output").Activate
Set Rng_Header = Range("B2:K2")
Set Rng_Body = Range("B3:K10")
Call z_Formatting_OutputTemplate("Output", Rng_Header, Rng_Body)
'select cell A1
Sheets("Output").Cells(1, 1).Select
'zoom to 80%
Sheets("Output").Activate
ActiveWindow.Zoom = 80

'***Workbook
'hide sheets
Sheets("Master Plan").Visible = False
Sheets("QualityCheckTkRsSorted").Visible = False
Sheets("QualityCheckPiRsSorted").Visible = False
'protect code
Call LockVBAProject(wb_new, "dqcode")

'check on the nof errors in the tool
Dim NofErrors As Long
NofErrors = z_RowSize(13, "Input") - 1
If NofErrors <> NofPiErrors + NofTkErrors + NofRsErrors Then
    Stop
End If

'set cursor
Sheets("Input").Activate
Sheets("Input").Range(Cells(11, 5), Cells(12, 7)).Select

End If
End Sub

Sub m_StartAll_Main_GenerateRecipientEmailList()
'pw for code: dqcode
Dim wbdate As String
wbdate = "2012_10_16"
Call m_Main_GenerateRecipientEmailList(wbdate)
End Sub

Sub m_Main_GenerateRecipientEmailList(Optional wbdate As String)
Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
    Array("Main_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))
'*****

Dim flag As Integer
'Dim wbdate As String
Dim reportversion_folder As String
Dim reportversion_file As String

```

```

*****

'reopen
flag = 1
flag_OpenExistingWbs = 1
If wdate = Empty Then
    wdate = "2012_10_16" 'VBA.DateTime.Year(Now()) & "_" & VBA.DateTime.Month(Now()) & "_"
& VBA.DateTime.Day(Now()) "'2012_2_8"
End If
reportversion_folder = "_V1-0"
reportversion_file = "_V1-0"
folderdescription = ""
*****

'Open and activate an Excel workbook (and session)
Dim WbPath As String
Dim WbName As String
Dim wb_new As Workbook
Dim Wb_DataQualityTool As Workbook
Dim Wb_CommunicationLists As Workbook
*****
*****
*****

'Adds the new workbook RD_MasterDataSet
*****
*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_DataQualityRecipientEmailList_" &
wdate & reportversion_file & ".xlsb" 'for test purposes
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, wb_new)
End If
    'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_DataQualityTool_" & wdate &
reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_DataQualityTool)
    'seems to help to protect an error
    Call z_WorkbookSave(Wb_DataQualityTool)
End If
'Open existing workbook
If flag_OpenExistingWbs Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
    WbName = "CommunicationLists.xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, Wb_CommunicationLists)
    'seems to help to protect an error
    Call z_WorkbookSave(Wb_CommunicationLists)
End If
'copy a sheet from another workbook
If flag_OpenExistingWbs Then

```

```

    Call z_CopySheetFromWb1ToWb2("Input", Wb_DataQualityTool, wb_new, "Begin")
    Call z_CopySheetFromWb1ToWb2("Outlook 2012_08_16", Wb_CommunicationLists, wb_new,
"Begin")
    Call z_CopySheetFromWb1ToWb2("PFM, PIM, PIC", Wb_CommunicationLists, wb_new, "Begin")
    Call z_CopySheetFromWb1ToWb2("Recipients Data Quality Tool", Wb_CommunicationLists,
wb_new, "Begin")
    Wb_CommunicationLists.Close False
    Wb_DataQualityTool.Close False
End If
If flag_OpenExistingWbs Then
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", wb_new)
        Call z_ShDelete("Sheet2", wb_new)
        Call z_ShDelete("Sheet3", wb_new)
    On Error GoTo 0
    Call z_WorkbookSave(wb_new)
End If
    'fill the sheet "Recipients Data Quality Tool" with data
    'recipient names from sheet input column B
If flag Then
    Sheets("Input").Activate
    Sheets("Input").Columns(2).Hidden = False
    Dim RowSize_from As Long
    RowSize_from = z_RowSize(2, "Input")
    Dim Rng_from As Range
    Set Rng_from = Sheets("Input").Range(Sheets("Input").Cells(2, 2),
Sheets("Input").Cells(RowSize_from, 2))
    Rng_from.Select
    Call z_CopyRange("Values", "Input", Rng_from, "Recipients Data Quality Tool", Sheets("Recipients
Data Quality Tool").Cells(2, 1))
End If
    'start mapping from PMEC List
If flag Then
    Call z_RecipientEmailAssignment_fromPMECList("Recipients Data Quality Tool", "PFM, PIM, PIC")
End If
    'start mapping from Outlook List
If flag Then
    Call z_RecipientEmailAssignment_FromOutlookList("Recipients Data Quality Tool", "Outlook
2012_08_16")
End If
    'delete the sheets
If flag Then
    Call z_ShDelete("Input", wb_new)
    Call z_ShDelete("Outlook 2012_08_16", wb_new)
    Call z_ShDelete("PFM, PIM, PIC", wb_new)
End If
    Call z_WorkbookSave(wb_new)
    wb_new.Close True
    MsgBox ("email the list to Sebastian")
End Sub

```



'new version with additional comparison

\*\*\*\*\*

Sub StartAll\_WorkbooksForTeamspace()

Dim wbdate As String

wbdate = "2012\_10\_16"

Call m\_Main\_WorkbooksForTeamspace(wbdate)

End Sub

Sub m\_Main\_WorkbooksForTeamspace(Optional wbdate As String)

Call z\_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", \_

Array("Main\_CheckBusinessRules", "Begin", "task:", "file:", CStr(Now()), ""))

\*\*\*\*\*

Dim flag As Integer

'Dim wbdate As String

Dim reportversion\_folder As String

Dim reportversion\_file As String

\*\*\*\*\*

'reopen

flag = 1

If wbdate = Empty Then

wbdate = "2012\_10\_16" 'VBA.DateTime.Year(Now()) & "\_" & VBA.DateTime.Month(Now()) & "\_"  
& VBA.DateTime.Day(Now()) "'2012\_2\_8"

End If

reportversion\_folder = "\_V1-0"

reportversion\_file = "\_V1-0"

folderdescription = ""

\*\*\*\*\*

'Open and activate an Excel workbook (and session)

Dim WbPath As String

Dim WbName As String

Dim wb\_new As Workbook

Dim wb\_old As Workbook

Dim Wb\_FilterRules As Workbook

Dim Wb\_AllIds As Workbook

Dim Wb\_OnLinePerRs As Workbook

Dim Wb\_CurrentAndBaseline As Workbook

\*\*\*\*\*

\*\*\*\*\*

'Adds the new workbook RD\_MasterDataSet

\*\*\*\*\*

\*\*\*\*\*

'FunctionalReporting

'=====

'document type: under evaluation, red flags o.k.

'document type: functional reporting, quality checks below thresholds or after manual adjustments

If flag Then

WbPath = "C:\Users\t740698\Desktop\PMEC\_VBA\_Master-DataSet\" & wbdate &  
reportversion\_folder & folderdescription & "\"

WbName = "CONFIDENTIAL\_SmC\_PMEC\_VBA\_MasterDataSet\_PivotFlat\_ForFunctionalReporting\_"  
& wbdate & reportversion\_file & ".xlsb"

Call z\_OpenAndActivateWb(WbName, WbPath, wb\_old)

End If

If flag Then

```

WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\ForTeamSpace" & "\"
WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_FunctionalReporting_" & wdate &
".xlsb"
Call z_MkDirIfNotExistent(WbPath)
Call z_WorkbookSaveAs(WbPath, WbName, wb_old, wb_new)
Call z_WorkbookSave(wb_new)
wb_new.Close True
End If
'PortfolioReporting
'=====
'document type: under evaluation, all unapproved and/or older versions
'document type: portfolio reporting, only quarterly reports approved from Heads RD Portfolio
If flag Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\
ZZZ_Inputs_MainGenerateReport\"
WbName = "DataFilteringRules" & ".xlsb"
Call z_OpenAndActivateWb(WbName, WbPath, Wb_FilterRules)
End If
If flag Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_ForPortfolioReportingFilterOut_" &
wdate & reportversion_file & ".xlsb"
Call z_OpenAndActivateWb(WbName, WbPath, wb_old)
End If
If flag Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\"
WbName =
"CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_ForPortfolioReportingAddBaseline_" & wdate &
reportversion_file & ".xlsb"
Call z_OpenAndActivateWb(WbName, WbPath, Wb_CurrentAndBaseline)
End If
If flag Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wdate &
reportversion_folder & folderdescription & "\ForTeamSpace" & "\"
WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PortfolioReporting_" & wdate &
".xlsb"
Call z_MkDirIfNotExistent(WbPath)
Call z_WorkbookSaveAs(WbPath, WbName, wb_old, wb_new)
'delete and copy sheets
On Error Resume Next
Call z_ShDelete("SmC_MasterDataSet_PiLevel", wb_new)
Call z_ShDelete("AdvancedFilters", wb_new)
Call z_ShDelete("Rem GlobSup&MS&CGM", wb_new)
Call z_ShDelete("CPD NEW AI_c", wb_new)
Call z_ShDelete("CPD NOT NEW AI_c", wb_new)
Call z_ShDelete("CPR CTD_c", wb_new)
Call z_ShDelete("CPR NOT CTD_c", wb_new)
Call z_ShDelete("CP highest lev_c", wb_new)
Call z_ShDelete("Seeds & BusinessDev_c", wb_new)
Call z_ShDelete("SmC_MasterDataSet_PiLevel_temp", wb_new)

```

```

On Error GoTo 0
Call z_CopySheetFromWb1ToWb2("Approval - Business Rules", Wb_FilterRules, wb_new, "End")
Call z_CopySheetFromWb1ToWb2("Comp_Current_Baseline", Wb_CurrentAndBaseline, wb_new,
"End")
Call z_CopySheetFromWb1ToWb2("Comp_Current_Retired", Wb_CurrentAndBaseline, wb_new,
"End")
Wb_FilterRules.Close False
Wb_CurrentAndBaseline.Close False
'find and replace attribute names
Dim RZx As String
'MsgBox ("Set the RZx")
'Stop
RZx = z_ReportVersionName(wbdate) ""R2x"
Dim Rng As Range
Dim ColSize As Long
ColSize = z_ColSize(1, "Comp_Current_Baseline")
Set Rng = Sheets("Comp_Current_Baseline").Range(Sheets("Comp_Current_Baseline").Cells(1, 1),
Sheets("Comp_Current_Baseline").Cells(1, ColSize))
Call z_FindAndReplace("Comp_Current_Baseline", Rng, "xlWhole", "PiIdentifier_", "PiIdentifier_" &
RZx & "andR0")
Call z_FindAndReplace("Comp_Current_Baseline", Rng, "xlPart", "_CurrentDataSet", "_" & RZx)
Call z_FindAndReplace("Comp_Current_Baseline", Rng, "xlPart", "_Baseline", "_R0")
Call z_FindAndReplace("Comp_Current_Baseline", Rng, "xlPart", "_c_", "_")
ColSize = z_ColSize(1, "SmC_MasterDataSet_PiLevel_fltr")
Set Rng =
Sheets("SmC_MasterDataSet_PiLevel_fltr").Range(Sheets("SmC_MasterDataSet_PiLevel_fltr").Cells(1
, 1), Sheets("SmC_MasterDataSet_PiLevel_fltr").Cells(1, ColSize))
Call z_FindAndReplace("SmC_MasterDataSet_PiLevel_fltr", Rng, "xlPart", "_c_", "")
'add pivot table
Dim FirstYr As Integer
FirstYr = VBA.DateTimes.year(Now()) - 1
Call z_AddPivotTable_LightVersion("Comp_Current_Baseline", "PivotTable_OneLinePerPI",
"PiIdentifier_" & RZx & "andR0", , "EAC Full Costs " & CStr(FirstYr + 1) & "_" & RZx, "EAC Full Costs " &
CStr(FirstYr + 1) & "_R0")
Dim StampDate As String
StampDate = z_StampDate(wbdate)
Call z_AddTimeStamp("PivotTable_OneLinePerPI", "05-Mar-2012 (R0) and " & StampDate & " (" &
RZx & ")", 1, 1, 25)
Call z_WorkbookSave(wb_new)
'add warning to pivots
Dim Text_warning As String
Text_warning = "Warning: Do only use Business Case figures in the Row Labels field of the Pivot but
not in the Values field"
If Sheets("PivotTable_OneLinePerPI").Cells(1, 1) Like "Data as*" Then
Call z_AddWarningText("PivotTable_OneLinePerPI", Text_warning, 2, 1, 25)
Else
Call z_AddWarningText("PivotTable_OneLinePerPI", Text_warning, 1, 1, 25)
End If
If Sheets("PivotTable_PiLevel_fltr").Cells(1, 1) Like "Data as*" Then
Call z_AddWarningText("PivotTable_PiLevel_fltr", Text_warning, 2, 1, 25)
Else
Call z_AddWarningText("PivotTable_PiLevel_fltr", Text_warning, 1, 1, 25)
End If

```

```

'refresh pivots
Call z_RefreshAllWorksheetPivots("PivotTable_OneLinePerPI")
Call z_RefreshAllWorksheetPivots("PivotTable_PiLevel_fltr")
'add EAC (by removing _c from the attribute name, the attribute was removed from the value field
after refresh)
Call z_Pivot_AddAttribute("PivotTable_PiLevel_fltr", "Value", "EAC Full Costs " & CStr(FirstYr + 1) &
"*, "Like")
'hide sheets
Sheets("Comp_Current_Baseline").Visible = False 'xlSheetVeryHidden removes it from the list
Sheets("SmC_MasterDataSet_PiLevel_fltr").Visible = False
'renaming of sheets
Sheets("Comp_Current_Baseline").name = "Comp " & RZx & " vs R0 SmCMasterDataSet"
Sheets("SmC_MasterDataSet_PiLevel_fltr").name = RZx & " SmC_MasterDataSet"
Sheets("PivotTable_OneLinePerPI").name = "Comparison " & RZx & " vs R0 Pivot"
Sheets("PivotTable_PiLevel_fltr").name = RZx & " Pivot"
*****

ColSize = z_ColSize(1, "Comp_Current_Retired")
Set Rng = Sheets("Comp_Current_Retired").Range(Sheets("Comp_Current_Retired").Cells(1, 1),
Sheets("Comp_Current_Retired").Cells(1, ColSize))
Call z_FindAndReplace("Comp_Current_Retired", Rng, "xlWhole", "PIdentifier_", "PIdentifier_" &
RZx & "andR2")
Call z_FindAndReplace("Comp_Current_Retired", Rng, "xlPart", "_CurrentDataSet", "_" & RZx)
Call z_FindAndReplace("Comp_Current_Retired", Rng, "xlPart", "_Baseline", "_R2")
Call z_FindAndReplace("Comp_Current_Retired", Rng, "xlPart", "_c_", "_")
Call z_AddPivotTable_LightVersion("Comp_Current_Retired", "PivotTable_OneLinePerPI",
"PIdentifier_" & RZx & "andR2", , "EAC Full Costs " & CStr(FirstYr + 1) & "_" & RZx, "EAC Full Costs " &
CStr(FirstYr + 1) & "_R2")
StampDate = z_StampDate("2012_05_31")
Call z_AddTimeStamp("PivotTable_OneLinePerPI", "31-May-2012 (R2) and " & StampDate & " (" &
RZx & ")", 1, 1, 25)
Call z_WorkbookSave(wb_new)
If Sheets("PivotTable_OneLinePerPI").Cells(1, 1) Like "Data as*" Then
Call z_AddWarningText("PivotTable_OneLinePerPI", Text_warning, 2, 1, 25)
Else
Call z_AddWarningText("PivotTable_OneLinePerPI", Text_warning, 1, 1, 25)
End If
Call z_RefreshAllWorksheetPivots("PivotTable_OneLinePerPI")
Sheets("Comp_Current_Retired").Visible = False
Sheets("Comp_Current_Retired").name = "Comp " & RZx & " vs R2 SmCMasterDataSet"
Sheets("PivotTable_OneLinePerPI").name = "Comparison " & RZx & " vs R2 Pivot"
wb_new.Close True
End If
'Milestone Reporting
'=====
'document type: milestones
If flag Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Milestones_" & wbdate &
reportversion_file & ".xlsb"
Call z_OpenAndActivateWb(WbName, WbPath, wb_old)
End If
If flag Then

```

```

WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\ForTeamSpace" & "\"
WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_Milestones_" & wbdate & ".xlsb"
Call z_MkDirIfNotExistent(WbPath)
Call z_WorkbookSaveAs(WbPath, WbName, wb_old, wb_new)
On Error Resume Next
Call z_ShDelete("PMEC_VBA_MasterDataSet_4", wb_new)
Call z_ShDelete("SmC_MasterDataSet_PiLevel", wb_new)
Call z_ShDelete("SmC_MasterDataSet_Milestones_1", wb_new)
Sheets("SmC_MasterDataSet_Milestones_2").name = "SmC_MasterDataSet_Milestones"
On Error GoTo 0
Call z_WorkbookSave(wb_new)
wb_new.Close True
End If
'Aggregated data sets
'=====
'document type: Aggregated Versions
If flag Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_AllIds_" & wbdate &
reportversion_file & ".xlsb"
Call z_OpenAndActivateWb(WbName, WbPath, Wb_AllIds)
End If
If flag Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"
WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForOneLinePerId_" &
wbdate & reportversion_file & ".xlsb"
Call z_OpenAndActivateWb(WbName, WbPath, wb_old)
End If
If flag Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\ForTeamSpace" & "\"
WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_AggregatedVersions_" & wbdate &
".xlsb"
Call z_MkDirIfNotExistent(WbPath)
Call z_WorkbookSaveAs(WbPath, WbName, wb_old, wb_new)
On Error Resume Next
Call z_ShDelete("PMEC_VBA_MasterDataSet_13", wb_new)
Call z_ShDelete("Pivot_PIs", wb_new)
Call z_ShDelete("Pivot_TKs", wb_new)
Call z_ShDelete("PivotTable_PiLevel", wb_new)
Call z_ShDelete("PivotTable_TkLevel", wb_new)
On Error GoTo 0
Call z_CopySheetFromWb1ToWb2("SmC_FullDataSet_PiLevel", Wb_AllIds, wb_new, "End")
Call z_CopySheetFromWb1ToWb2("SmC_FullDataSet_TkLevel", Wb_AllIds, wb_new, "End")
Wb_AllIds.Close False
End If
If flag Then
WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"

```

```

WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_PivotFlat_ForOneLinePerId2_" &
wbdate & reportversion_file & ".xlsb"
Call z_OpenAndActivateWb(WbName, WbPath, Wb_OnLinePerRs)
End If
If flag Then
    Call z_CopySheetFromWb1ToWb2("SmC_MasterDataSet_ResourceLevel", Wb_OnLinePerRs,
wb_new, "End")
End If
If flag Then
    Wb_OnLinePerRs.Close False
    Call z_WorkbookSave(wb_new)
End If
If flag Then
    wb_new.Activate
    'move tabs
    Sheets("SmC_FullDataSet_TkLevel").Move Before:=Sheets(1)
    Sheets("SmC_FullDataSet_PiLevel").Move Before:=Sheets(2)
    'color tabs
    Sheets("SmC_FullDataSet_TkLevel").Tab.Color = 5296274
    Sheets("SmC_FullDataSet_PiLevel").Tab.Color = 5296274
    Sheets("SmC_MasterDataSet_ResourceLevel").Tab.Color = 15773696
    Sheets("SmC_MasterDataSet_TkLevel").Tab.Color = 15773696
    Sheets("SmC_MasterDataSet_PiLevel").Tab.Color = 15773696
    'timestamps
    StampDate = z_StampDate(wbdate)
    Call z_AddTimeStamp("SmC_FullDataSet_TkLevel", StampDate, 1, 1, 25)
    Call z_AddTimeStamp("SmC_FullDataSet_PiLevel", StampDate, 1, 1, 25)
    'SmC_FullDataSet_TkLevel
    Sheets("SmC_FullDataSet_TkLevel").Activate
    Dim col_PiHasRes As Long
    Dim col_MD4 As Long
    col_PiHasRes = z_GetColumnIndex("Pi has Resource(s)", 1, "SmC_FullDataSet_TkLevel")
    Call z_DeleteColumns("SmC_FullDataSet_TkLevel", col_PiHasRes, col_PiHasRes)
    col_MD4 = z_GetColumnIndex("TkIdentifier from PMEC_VBA_MasterDataSet_4", 1,
"SmC_FullDataSet_TkLevel")
    Call z_DeleteColumns("SmC_FullDataSet_TkLevel", col_MD4, col_MD4)

    Dim col_WsId As Long
    Dim col_ActivityId As Long
    Dim col_TkId As Long
    col_TkId = z_GetColumnIndex("TkIdentifier", 1, "SmC_FullDataSet_TkLevel")
    col_WsId = z_GetColumnIndex("WsIdentifier", 1, "SmC_FullDataSet_TkLevel")
    col_ActivityId = z_GetColumnIndex("ActivityIdentifier", 1, "SmC_FullDataSet_TkLevel")
    Call z_MoveColumn("SmC_FullDataSet_TkLevel", col_WsId, "SmC_FullDataSet_TkLevel", col_TkId +
1)
    Call z_MoveColumn("SmC_FullDataSet_TkLevel", col_ActivityId, "SmC_FullDataSet_TkLevel",
col_TkId + 2)

    Dim col_TkHasRes As Long
    Dim col_IncludedInMasterDS As Long
    col_IncludedInMasterDS = z_GetColumnIndex("TkIdentifier from SmC_MasterDataSet_TkLevel", 1,
"SmC_FullDataSet_TkLevel")
    Cells(1, col_IncludedInMasterDS).Value = "Included in MasterDataSet"

```

```

col_IncludedInMasterDS = z_GetColumnIndex("Included in MasterDataSet", 1,
"SmC_FullDataSet_TkLevel")
col_TkHasRes = z_GetColumnIndex("Task has Resource(s)", 1, "SmC_FullDataSet_TkLevel")
Dim RowSize_Tk As Long
RowSize_Tk = z_RowSize(col_TkId, "SmC_FullDataSet_TkLevel")
For Row = 2 To RowSize_Tk
    If Cells(Row, col_TkId) <> "(blank)" Then
        If Cells(Row, col_TkHasRes) = "(blank)" Then
            Cells(Row, col_TkHasRes) = "TRUE"
        End If

        If Cells(Row, col_IncludedInMasterDS) = "(blank)" Then
            Cells(Row, col_IncludedInMasterDS) = "FALSE"
        Else
            Cells(Row, col_IncludedInMasterDS) = "TRUE"
        End If
    End If
Next

'SmC_FullDataSet_TkLevel
Sheets("SmC_FullDataSet_PiLevel").Activate
Dim RowSize_Pi As Long
Dim Col_PiId As Long
col_IncludedInMasterDS = z_GetColumnIndex("Pidentifier from SmC_MasterDataSet_PiLevel", 1,
"SmC_FullDataSet_PiLevel")
Cells(1, col_IncludedInMasterDS).Value = "Included in MasterDataSet"
col_IncludedInMasterDS = z_GetColumnIndex("Included in MasterDataSet", 1,
"SmC_FullDataSet_PiLevel")
Col_PiId = z_GetColumnIndex("Pidentifier", 1, "SmC_FullDataSet_PiLevel")
col_PiHasRes = z_GetColumnIndex("Pi has Resource(s)", 1, "SmC_FullDataSet_PiLevel")
RowSize_Pi = z_RowSize(Col_PiId, "SmC_FullDataSet_PiLevel")
For Row = 2 To RowSize_Pi
    If Cells(Row, col_PiHasRes) = "(blank)" Then
        Cells(Row, col_PiHasRes) = "TRUE"
    End If

    If Cells(Row, col_IncludedInMasterDS) = "(blank)" Then
        Cells(Row, col_IncludedInMasterDS) = "FALSE"
    Else
        Cells(Row, col_IncludedInMasterDS) = "TRUE"
    End If
Next
End If
If flag Then
    Call z_WorkbookSave(wb_new)
    wb_new.Close True
End If

'Data Quality Tool
'=====
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\"

```

```

    WbName = "CONFIDENTIAL_SmC_PMEC_VBA_MasterDataSet_DataQualityTool_" & wbdate &
reportversion_file & ".xlsb"
    Call z_OpenAndActivateWb(WbName, WbPath, wb_old)
End If
If flag Then
    WbPath = "C:\Users\t740698\Desktop\PMEC_VBA_Master-DataSet\" & wbdate &
reportversion_folder & folderdescription & "\ForTeamSpace" & "\"
    WbName = "SmC_Data_Quality_Tool.xlsb"
    Call z_MkDirIfNotExistent(WbPath)
    Call z_WorkbookSaveAs(WbPath, WbName, wb_old, wb_new)
End If
End Sub

```

```

Sub m_Main_Strawman()
    answer = InputBox("R3 SmC_MasterDataSet aktiviert?", , "YES", 13500, 12500)
    If answer <> "YES" Then
        Stop
    End If
    answer = InputBox("New Dimensions already added?", , "YES", 13500, 12500)
    If answer = "YES" Then
        flag = 0
    Else
        flag = 1
    End If

    Dim Sh As String
    Sh = "R3 SmC_MasterDataSet"
    Sheets(Sh).Activate
    Dim Col_Pild As Long
    Dim col_PISubType As Long
    Dim col_PIStage As Long
    Dim Col_PL1 As Long
    Dim col_PL2 As Long
    Dim col_PL3 As Long
    Dim Col_StrategicCrop As Long
    Col_Pild = z_GetColumnIndex("Pildentifier", 1, Sh)
    col_PISubType = z_GetColumnIndex("PI Sub Type", 1, Sh)
    col_PIStage = z_GetColumnIndex("PI Stage", 1, Sh)
    Col_PL1 = z_GetColumnIndex("Portfolio Level 1", 1, Sh)
    col_PL2 = z_GetColumnIndex("Portfolio Level 2", 1, Sh)
    col_PL3 = z_GetColumnIndex("Portfolio Level 3", 1, Sh)
    Col_StrategicCrop = z_GetColumnIndex("PI Strategic Crop", 1, Sh)
    Dim RowSize As Long
    RowSize = z_RowSize(Col_Pild, Sh)
    'add new dimensions/columns
    If flag Then
        Dim Col_Start As Long
        Col_Start = z_GetColumnIndex("PI Strategic Crop", 1, Sh)
        Call z_InsertEmptyCols(Sh, 3, Col_Start)
        Call z_AddColNames(Array("Technology", "InnovationLifeCycle", "InvestmentSegment"), Sh, 1,
Col_Start, , "No")
    End If

```



'fill the new dimensions/columns

\*\*\*\*\*

Dim col\_ICS As Long

Dim col\_BC As Long

Dim col\_CE As Long

Dim col\_PER As Long

Dim col\_AT As Long

col\_ICS = z\_GetColumnIndex("ICS? YES/NO", 1, Sh)

col\_BC = z\_GetColumnIndex("BIO-CONTROLS? YES/NO", 1, Sh)

col\_CE = z\_GetColumnIndex("CE? YES/NO", 1, Sh)

col\_PER = z\_GetColumnIndex("PER? YES/NO", 1, Sh)

col\_AT = z\_GetColumnIndex("Adjacent Technology YES/NO", 1, Sh)

Dim col\_Technology As Long

col\_Technology = z\_GetColumnIndex("Technology", 1, Sh)

Dim col\_InnovationLifeCyle As Long

col\_InnovationLifeCyle = z\_GetColumnIndex("InnovationLifeCycle", 1, Sh)

Dim Col\_InvestmentSegment As Long

Col\_InvestmentSegment = z\_GetColumnIndex("InvestmentSegment", 1, Sh)

'Genetics

\*\*\*\*\*

flag = 1

If flag Then

For Row = 2 To RowSize

If LCase(Cells(Row, col\_ICS)) = "no" \_

And LCase(Cells(Row, col\_AT)) = "no" \_

And LCase(Cells(Row, col\_BC)) = "no" \_

And LCase(Cells(Row, col\_CE)) = "no" \_

And LCase(Cells(Row, col\_PER)) = "no" \_

Then

If Cells(Row, col\_PISubType) = "Breeding" \_

Or Cells(Row, col\_PISubType) = "GM Trait" \_

Or Cells(Row, col\_PISubType) = "GM Trait Discontinuation" \_

Or Cells(Row, col\_PISubType) = "GM Trait Stack" \_

Or Cells(Row, col\_PISubType) = "GM Trait Stack Extension" \_

Or Cells(Row, col\_PISubType) = "Native Trait" \_

Or Cells(Row, col\_PISubType) Like "Capability and Technology\*" \_

Then

If Cells(Row, Col\_PL1) <> "CROP\_PROTECTION" \_

Then

'Technology

\*\*\*\*\*

'Cells(Row, col\_Technology).Activate

Cells(Row, col\_Technology) = "1.Genetics"

'InvestmentSegment

\*\*\*\*\*

If Cells(Row, col\_PISubType) = "GM Trait" \_

Or Cells(Row, col\_PISubType) = "GM Trait Discontinuation" \_

Or Cells(Row, col\_PISubType) = "GM Trait Stack" \_

Or Cells(Row, col\_PISubType) = "GM Trait Stack Extension" \_

```

    Then
        Cells(Row, Col_InvestmentSegment) = "1.a.Genetic Modification Trait"
    End If
    If Cells(Row, col_PISubType) = "Native Trait" Then
        Cells(Row, Col_InvestmentSegment) = "1.b.Native Traits"
    End If
    If Cells(Row, col_PISubType) = "Breeding" _
        Then
            Cells(Row, Col_InvestmentSegment) = "1.c.Breeding"
        End If

'this rule may be false (only implemented to get the same numbers)
    If Cells(Row, col_PISubType) Like "Capability and Technology*" Then
        If Cells(Row, col_PISubType) = "Seeds-1" _
            Or Cells(Row, col_PISubType) = "Seeds-2" _
            Or Cells(Row, col_PISubType) = "Seeds-3" _
            Or Cells(Row, col_PISubType) = "Seeds-4" _
            Or Cells(Row, col_PISubType) = "Seeds-5" _
            Or Cells(Row, col_PISubType) = "Seeds-6" _
            Or Cells(Row, col_PISubType) = "Seeds-7" _
            Or Cells(Row, col_PISubType) = "Seeds-8" _
            Or Cells(Row, col_PISubType) = "Seeds-9" _
            Or Cells(Row, col_PISubType) = "Seeds-10" _
            Or Cells(Row, col_PISubType) = "Seeds-11" _
            Then
                Cells(Row, Col_InvestmentSegment) = "1.c.Breeding"
            Else
                Cells(Row, Col_InvestmentSegment) = "1.a.Genetic Modification Trait"
            End If
        End If

'InnovationLifeCycle
'*****

'Genetics-Research
'-----
    If Cells(Row, col_PISubType) = "Seeds-1" _
        Or Cells(Row, col_PISubType) = "Seeds-2" _
        Or Cells(Row, col_PISubType) = "Seeds-3" _
        Or Cells(Row, col_PISubType) = "Seeds-Discovery" _
        Or Cells(Row, col_PISubType) = "Seeds-Proof of Concept" _
        Or Cells(Row, col_PISubType) = "(blank)" _
        Then
            Cells(Row, col_InnovationLifeCycle) = "1.1.Genetics - Research"
        End If

'Genetics-NewProducts&Extensions
'-----
    If Cells(Row, col_PISubType) = "Seeds-4" _
        Or Cells(Row, col_PISubType) = "Seeds-5" _
        Or Cells(Row, col_PISubType) = "Seeds-6" _
        Or Cells(Row, col_PISubType) = "Seeds-Early Development" _
        Or Cells(Row, col_PISubType) = "Seeds-Late Development" _
        Or Cells(Row, col_PISubType) = "Seeds-Pre-Commercial" _
        Then

```

```

        Cells(Row, col_InnovationLifeCycle) = "1.2.Genetics - New Products & Extensions"
    End If
    'Genetics-ProductMaintenance
    '-----
    If Cells(Row, col_PIS tage) = "Seeds-7" _
        Or Cells(Row, col_PIS tage) = "Seeds-8" _
        Or Cells(Row, col_PIS tage) = "Seeds-9" _
        Or Cells(Row, col_PIS tage) = "Seeds-10" _
        Or Cells(Row, col_PIS tage) = "Seeds-11" _
        Or Cells(Row, col_PIS tage) = "Seeds-Commercial" _
        Or Cells(Row, col_PIS tage) = "Seeds-Discontinue" _
    Then
        Cells(Row, col_InnovationLifeCycle) = "1.3.Genetics - Product Maintenance"
    End If
End If
End If
End If
Next
End If

'New&IntegratedTechnology
'*****

If flag Then
    For Row = 2 To RowSize
        If LCase(Cells(Row, col_AT)) = "yes" Then
            'Technology
            '*****

            'Cells(Row, col_Technology).Activate
            Cells(Row, col_Technology) = "2.New & Integrated Technology"

            'InvestmentSegment
            '*****

            If LCase(Cells(Row, col_IC S)) = "yes" Then
                Cells(Row, Col_InvestmentSegment) = "2.a.Integrated Solutions"
            ElseIf LCase(Cells(Row, col_AT)) = "yes" Then
                Cells(Row, Col_InvestmentSegment) = "2.b.Adjacent Technology"
            ElseIf LCase(Cells(Row, col_BC)) = "yes" Then
                Cells(Row, Col_InvestmentSegment) = "2.c.Bio Controls"
            ElseIf LCase(Cells(Row, col_CE)) = "yes" Then
                Cells(Row, Col_InvestmentSegment) = "2.d.Crop Enhancement"
            End If

            'InnovationLifeCycle
            '*****

            'New&IntegratedTechnology-Research
            '-----

            If Cells(Row, Col_PiId) = "PI0009812" _
                Or Cells(Row, Col_PiId) = "PI0009852" _
            Then
                Cells(Row, col_InnovationLifeCycle) = "2.1.New & Integrated Technology - Research"

                'New&IntegratedTechnology-NewProducts&Extensions
                '-----

```

```

Else
    Cells(Row, col_InnovationLifeCyle) = "2.2.New & Integrated Technology - New Product &
Solution Development"
End If
End If
If (LCase(Cells(Row, col_ICS)) = "yes" _
Or LCase(Cells(Row, col_BC)) = "yes" _
Or LCase(Cells(Row, col_CE)) = "yes") _
And LCase(Cells(Row, col_AT)) = "no" _
And LCase(Cells(Row, col_PER)) = "no" _
Then

'Technology
*****

'Cells(Row, col_Technology).Activate
Cells(Row, col_Technology) = "2.New & Integrated Technology"

'InvestmentSegment
*****

If LCase(Cells(Row, col_ICS)) = "yes" Then
    Cells(Row, Col_InvestmentSegment) = "2.a.Integrated Solutions"
ElseIf LCase(Cells(Row, col_AT)) = "yes" Then
    Cells(Row, Col_InvestmentSegment) = "2.b.Adjacent Technology"
ElseIf LCase(Cells(Row, col_BC)) = "yes" Then
    Cells(Row, Col_InvestmentSegment) = "2.c.Bio Controls"
ElseIf LCase(Cells(Row, col_CE)) = "yes" Then
    Cells(Row, Col_InvestmentSegment) = "2.d.Crop Enhancement"
End If

'InnovationLifeCycle
*****

'New&IntegratedTechnology-Research
'-----
If Cells(Row, col_PIS tage) = "Seeds-1" _
Or Cells(Row, col_PIS tage) = "Seeds-2" _
Or Cells(Row, col_PIS tage) = "Seeds-3" _
Or Cells(Row, col_PIS tage) = "Seeds-Discovery" _
Or Cells(Row, col_PIS tage) = "Seeds-Proof of Concept" _
Or Cells(Row, col_PIS tage) = "CP-1-Research" _
Then
    Cells(Row, col_InnovationLifeCyle) = "2.1.New & Integrated Technology - Research"
End If
'New&IntegratedTechnology-NewProducts&Extensions
'-----
If Cells(Row, col_PIS tage) = "Seeds-4" _
Or Cells(Row, col_PIS tage) = "Seeds-5" _
Or Cells(Row, col_PIS tage) = "Seeds-6" _
Or Cells(Row, col_PIS tage) = "Seeds-Early Development" _
Or Cells(Row, col_PIS tage) = "Seeds-Late Development" _
Or Cells(Row, col_PIS tage) = "Seeds-Pre-Commercial" _
Or Cells(Row, col_PIS tage) = "Seeds-7" _
Or Cells(Row, col_PIS tage) = "Seeds-8" _
Or Cells(Row, col_PIS tage) = "Seeds-9" _

```

```

Or Cells(Row, col_PISStage) = "Seeds-10" _
Or Cells(Row, col_PISStage) = "Seeds-11" _
Or Cells(Row, col_PISStage) = "Seeds-Commercial" _
Or Cells(Row, col_PISStage) = "Seeds-Discontinue" _
Or Cells(Row, col_PISStage) = "CP-2-Evaluation" _
Or Cells(Row, col_PISStage) = "CP-3-Development" _
Or Cells(Row, col_PISStage) = "CP-4-Life Cycle Mgmt" _
Or Cells(Row, col_PISStage) = "CP-A-Feasibility" _
Or Cells(Row, col_PISStage) = "CP-B-Evaluation" _
Or Cells(Row, col_PISStage) = "CP-C-Development" _
Or Cells(Row, col_PISStage) = "CP-D-Sales" _
Or Cells(Row, col_PISStage) = "CP-Not Applicable" _
Then
Cells(Row, col_InnovationLifeCyle) = "2.2.New & Integrated Technology - New Product &
Solution Development"
End If
End If
Next
End If

```

'Chemicals

\*\*\*\*\*

```

If flag Then
For Row = 2 To RowSize
'all PER
If LCase(Cells(Row, col_PER)) = "yes" Then
'Technology
*****

'Cells(Row, col_Technology).Activate
Cells(Row, col_Technology) = "3.Chemicals"
'InvestmentSegment
*****

If Cells(Row, col_PL3) = "CPD_SEED_TREATMENT" _
Or Cells(Row, col_PL3) = "CPR_SEED_TREATMENT" _
Then
Cells(Row, Col_InvestmentSegment) = "3.a.Seed Care"
End If
If Cells(Row, col_PL3) = "CPD_HERBICIDES" _
Or Cells(Row, col_PL3) = "CPR_HERBICIDES" _
Then
Cells(Row, Col_InvestmentSegment) = "3.b.Herbicides"
End If
If Cells(Row, col_PL3) = "CPD_FUNGICIDES" _
Or Cells(Row, col_PL3) = "CPR_FUNGICIDES" _
Then
Cells(Row, Col_InvestmentSegment) = "3.c.Fungicides"
End If
If Cells(Row, col_PL3) = "CPD_INSECTICIDES" _
Or Cells(Row, col_PL3) = "CPR_INSECTICIDES" _
Or Cells(Row, col_PL2) = "LAWN_GARDEN" _
Then
Cells(Row, Col_InvestmentSegment) = "3.d.Insecticides"
End If

```

```

'InnovationLifeCycle
*****

Cells(Row, col_InnovationLifeCyle) = "3.1.Chemicals - Research Incl. PER"
End If
,

If LCase(Cells(Row, col_ICS)) = "no" _
And LCase(Cells(Row, col_AT)) = "no" _
And LCase(Cells(Row, col_BC)) = "no" _
And LCase(Cells(Row, col_CE)) = "no" _
And LCase(Cells(Row, col_PER)) = "no" _
Then
If Cells(Row, col_PISubType) = "AI New" _
Or Cells(Row, col_PISubType) = "Capability & Technology Development" _
Or Cells(Row, col_PISubType) Like "Capability and Technology*" _
Or Cells(Row, col_PISubType) = "Formulation Extension" _
Or Cells(Row, col_PISubType) = "Formulation New" _
Or Cells(Row, col_PISubType) = "Idea Evaluation" _
Or Cells(Row, col_PISubType) = "Label Extension" _
Or Cells(Row, col_PISubType) = "Non-Product Customer Offer" _
Or Cells(Row, col_PISubType) = "Pack Development" _
Or Cells(Row, col_PISubType) = "Unplanned Resource" _
Or Cells(Row, col_PISubType) = "AI & Product Maintenance" _
Or Cells(Row, col_PISubType) = "AI Re-registration" _
Or Cells(Row, col_PISubType) = "Stewardship" _
Then
If Cells(Row, col_PL2) <> "CROP_PROTECTION - Level 1 portfolio" _
And Cells(Row, col_PL3) <> "CPD - Level 2 portfolio" _
And Cells(Row, col_PL3) <> "CPR - Level 2 portfolio" _
And Cells(Row, col_PL3) <> "CPR_EPICC" _
Then
If Cells(Row, Col_PL1) <> "SEEDS" _
Then
'Technology
*****

'Cells(Row, col_Technology).Activate
Cells(Row, col_Technology) = "3.Chemicals"

'InvestmentSegment
*****

If Cells(Row, col_PL3) = "CPD_SEED_TREATMENT" _
Or Cells(Row, col_PL3) = "CPR_SEED_TREATMENT" _
Then
Cells(Row, Col_InvestmentSegment) = "3.a.Seed Care"
End If
If Cells(Row, col_PL3) = "CPD_HERBICIDES" _
Or Cells(Row, col_PL3) = "CPR_HERBICIDES" _
Then
Cells(Row, Col_InvestmentSegment) = "3.b.Herbicides"
End If
If Cells(Row, col_PL3) = "CPD_FUNGICIDES" _
Or Cells(Row, col_PL3) = "CPR_FUNGICIDES" _
Then
Cells(Row, Col_InvestmentSegment) = "3.c.Fungicides"

```

```

End If
If Cells(Row, col_PL3) = "CPD_INSECTICIDES" _
    Or Cells(Row, col_PL3) = "CPR_INSECTICIDES" _
    Or Cells(Row, col_PL2) = "LAWN_GARDEN" _
    Then
        Cells(Row, Col_InvestmentSegment) = "3.d.Insecticides"
    End If

'InnovationLifeCycle
*****

'Chemicals-ResearchInclPER
'-----
If Cells(Row, col_PISubType) = "AI New" Then
    If Cells(Row, col_PIStage) = "CP-1-Research" _
        Or Cells(Row, col_PIStage) = "CP-A-Feasibility" Then
        Cells(Row, col_InnovationLifeCycle) = "3.1.Chemicals - Research Incl. PER"
    End If
End If

'Chemicals-NewAIDevelopment
'-----
If Cells(Row, col_PISubType) = "AI New" Then
    If Cells(Row, col_PIStage) = "CP-2-Evaluation" _
        Or Cells(Row, col_PIStage) = "CP-3-Development" _
        Or Cells(Row, col_PIStage) = "CP-4-Life Cycle Mgmt" _
        Or Cells(Row, col_PIStage) = "CP-B-Evaluation" _
        Or Cells(Row, col_PIStage) = "CP-C-Development" _
        Or Cells(Row, col_PIStage) = "CP-D-Sales" _
        Or Cells(Row, col_PIStage) = "CP-Not Applicable" _
    Then
        Cells(Row, col_InnovationLifeCycle) = "3.2.Chemicals - New AI Development"
    End If
End If

'Chemicals-NewProducts&Extensions
'-----
If Cells(Row, col_PISubType) = "Capability & Technology Development" _
    Or Cells(Row, col_PISubType) Like "Capability and Technology*" _
    Or Cells(Row, col_PISubType) = "Formulation Extension" _
    Or Cells(Row, col_PISubType) = "Formulation New" _
    Or Cells(Row, col_PISubType) = "Idea Evaluation" _
    Or Cells(Row, col_PISubType) = "Label Extension" _
    Or Cells(Row, col_PISubType) = "Non-Product Customer Offer" _
    Or Cells(Row, col_PISubType) = "Pack Development" _
    Or Cells(Row, col_PISubType) = "Unplanned Resource" _
    Then
        Cells(Row, col_InnovationLifeCycle) = "3.3.Chemicals - New Products & Extensions"
    End If

'Chemicals-ProductMaintenance
'-----
If Cells(Row, col_PISubType) = "AI & Product Maintenance" _
    Or Cells(Row, col_PISubType) = "AI Re-registration" _
    Or Cells(Row, col_PISubType) = "Stewardship" _
    Then

```

```

        Cells(Row, col_InnovationLifeCyle) = "3.4.Chemicals - Product Maintenance"
    End If
End If
End If
End If
End If
Next
End If
'GlobalFunctions&Platforms
'*****

If flag Then
    For Row = 2 To RowSize
        If LCase(Cells(Row, col_ICS)) = "no" _
            And LCase(Cells(Row, col_AT)) = "no" _
            And LCase(Cells(Row, col_BC)) = "no" _
            And LCase(Cells(Row, col_CE)) = "no" _
            And LCase(Cells(Row, col_PER)) = "no" _
        Then

            'GlobalFunctions&Platforms-GlobalSupportFunctions
            '-----

            If Cells(Row, col_PISubType) = "Platforms | Global Functions" Then
                'Technology
                '*****

                'Cells(Row, col_Technology).Activate
                Cells(Row, col_Technology) = "4.Global Functions & Capabilities"
                'InnovationLifeCycle
                '*****

                Cells(Row, col_InnovationLifeCyle) = "4.1.Global Functions & Capabilities - Global Support
Functions"
            End If
            'GlobalFunctions&Platforms-RDIS
            '-----

            If Cells(Row, col_PISubType) = "Platforms | RD-IS" Then
                'Technology
                '*****

                'Cells(Row, col_Technology).Activate
                Cells(Row, col_Technology) = "4.Global Functions & Capabilities"
                'InnovationLifeCycle
                '*****

                Cells(Row, col_InnovationLifeCyle) = "4.2.Global Functions & Capabilities - R&D-IS"
            End If
            'GlobalFunctions&Platforms-CropSpecificPlatforms
            '-----

            If Cells(Row, col_PISubType) = "Platform" _
                Or Cells(Row, col_PISubType) = "Management" _
                Or Cells(Row, col_PISubType) = "Platforms | Management" _
                Or Cells(Row, col_PISubType) = "Platforms | Capability & Technology Development" _
            Then
                'Technology
                '*****

                'Cells(Row, col_Technology).Activate
                Cells(Row, col_Technology) = "4.Global Functions & Capabilities"
            End If
        End If
    Next
End If

```



```

    'InnovationLifeCycle
    *****

    Cells(Row, col_InnovationLifeCyle) = "4.3.Global Functions & Capabilities - Crop Specific
Platforms"
End If
If Cells(Row, col_PISubType) = "Platforms | Platform" _
And Cells(Row, Col_StrategicCrop) <> "Non-Crop" _
Then
    'Technology
    *****

    'Cells(Row, col_Technology).Activate
    Cells(Row, col_Technology) = "4.Global Functions & Capabilities"
    'InnovationLifeCycle
    *****

    Cells(Row, col_InnovationLifeCyle) = "4.3.Global Functions & Capabilities - Crop Specific
Platforms"
End If

'GlobalFunctions&Platforms-CrossCropPlatforms
'-----
If (Cells(Row, col_PL2) = "CROP_PROTECTION - Level 1 portfolio" _
Or Cells(Row, col_PL3) = "CPD - Level 2 portfolio" _
Or Cells(Row, col_PL3) = "CPR - Level 2 portfolio" _
Or Cells(Row, col_PL3) = "CPR_EPICC") _
And Cells(Row, col_PISubType) <> "Platform" _
And Cells(Row, col_PISubType) <> "Platforms | Platform" _
And Cells(Row, col_PISubType) <> "Management" _
And Cells(Row, col_PISubType) <> "Platforms | Management" _
And Cells(Row, col_PISubType) <> "Platforms | Capability & Technology Development" _
Then
    'Technology
    *****

    'Cells(Row, col_Technology).Activate
    Cells(Row, col_Technology) = "4.Global Functions & Capabilities"
    'InnovationLifeCycle
    *****

    If Cells(Row, col_PISubType) = "Platforms | Capability & Technology Development" Then
        Stop
    End If
    Cells(Row, col_InnovationLifeCyle) = "4.4.Global Functions & Capabilities - Cross Crop
Platforms"
End If
If Cells(Row, col_PISubType) = "Platforms | Platform" _
And Cells(Row, Col_StrategicCrop) = "Non-Crop" _
Then
    'Technology
    *****

    'Cells(Row, col_Technology).Activate
    Cells(Row, col_Technology) = "4.Global Functions & Capabilities"
    'InnovationLifeCycle
    *****

    Cells(Row, col_InnovationLifeCyle) = "4.4.Global Functions & Capabilities - Cross Crop
Platforms"

```

```

        End If

    End If
Next
End If

End Sub

```

## **File: DownloadScript\_v6.docm**

```

'Manual tasks/settings before starting the macro.
'Goto:(*Preparations before running the macro
'Tools>References
'Visual Basic For Applications
'Microsoft Excel 12.0 Object Library
'OLE Automation
'Microsoft Office 12.0 Object Library
'Microsoft Forms 2.0 Object Library
'Microsoft Visual Basic for Applications Extensibility 5.3c:/Program Files (x86)/Common
Files/Microsoft Shared/VBA/VBA6/VBE6EXT.OLB
'Microsoft IMAPI2 Base Functionality-c:/Windows/system32/imapi2.dll
'Microsoft Internet Controls-c:/Windows/SysWOW64/ieframe.dll

'(*Constant initialisation
Public Const SW_RESTORE = 9
Public Const SW_SHOW = 5
Public Const SW_MINIMIZE = 6
')*Constant initialisation

Private Sub m_Start_Main_DownloadReport_HTML_Export()
'Activity exported with 1997/2000 Export via HTML
'Pls exported with 2007/2010 Export via HTML
Dim xlapp As Excel.Application
Set xlapp = CreateObject("Excel.Application")
xlapp.Visible = True
Dim PlandActivity As String
Dim ResourceReportFlag As Integer
Dim myPageURL As String
Dim answer_1 As String
Dim answer_2 As String
Dim answer_3 As String
*****
answer_1 = InputBox("stg or pro", , "pro", 13500, 12500)
If answer_1 = "stg" Then
    myPageURL = "smartchoice.stg.intra"
Elseif answer_1 = "pro" Then
    myPageURL = "smartchoice.pro.intra"
Else
    Stop
End If

```

```

answer_2 = InputBox("Costs(Gant) = Yes; PlannedSales(Tracking) = No", , "Yes", 13500, 12500)
If answer_2 = "Yes" Then
    ResourceReportFlag = 2
    answer_3 = InputBox("PiAndActivity = 11; OnlyPI = 10; OnlyActivity=01", , "11", 13500, 12500)
    If answer_3 = "11" Then
        PlandActivity = "11"
    ElseIf answer_3 = "10" Then
        PlandActivity = "10"
    ElseIf answer_3 = "01" Then
        PlandActivity = "01"
    Else
        Stop
    End If
ElseIf answer_2 = "No" Then
    ResourceReportFlag = 3
    answer_3 = InputBox("PiAndActivity = 11; OnlyPI = 10; OnlyActivity=01", , "01", 13500, 12500)
    If answer_3 = "11" Then
        PlandActivity = "11"
    ElseIf answer_3 = "10" Then
        PlandActivity = "10"
    ElseIf answer_3 = "01" Then
        PlandActivity = "01"
    Else
        Stop
    End If
Else
    Stop
End If

xlapp.Wait (Now + TimeValue("0:00:02"))
Call m_Main_DownloadReport_HTML_Export(xlapp, PlandActivity, myPageURL,
ResourceReportFlag)
End Sub

Private Sub m_Main_DownloadReport_HTML_Export(ByRef xlapp As Excel.Application, _
    PlandActivity As String, myPageURL As String, ResourceReportFlag As Integer)
    'Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
        Array("Main_DownloadReport", "Begin", "task: ", CStr(Now()), ""))
'(*Variable declaration
'(*Preparations before running the macro
'Dim PlandActivity As Integer
Dim Sh_log As String
Dim delta1 As Integer
Dim delta2 As Integer
Dim mywb As Workbook
Dim WbPath As String
Dim wbdate As String
Dim WbName As String
'(*Create a new Excel Workbook
'(*prepare the SmC window in an IE object
Dim myIE As SHDocVw.InternetExplorer
Dim myPageTitle As String

```

```

'Dim myPageURL As String
Dim strWindowTitle As String 'used several times for different names
Dim My_IE_hWnd As Long
'(*Open the SmC Project-Module
Dim Target_X0 As Long
Dim Target_Y0 As Long
Dim BySyngentaPortfolio As Boolean
'(*ResetButton
Dim ResetButtonYes As Integer
'(*Resource Report
Dim VP_DropdownSecondPos As Long
Dim VP_Start As Integer
Dim TotalVPs As Integer
Dim FailedDownloads_iter As Integer
Dim VP_iter As Integer
'(**Open the OneLinePerResource Report
Dim ResourceStyle As Long
Dim ResourceStyleDelta As Long
'(**Workaround for the selection problem: clicks on the scroll bar
Dim ScrollClicks_iter As Integer
Dim ScrollClicks_iter_Max As Integer
'(**Excel Download
Dim sec As Integer
Dim DurationSinceDownloadClick As Integer
Dim secMax_Pi As Integer
Dim My_IE_XL_hWnd As Long
Dim My_IE_XL_Child_hWnd As Long
Dim MyChildName As String
Dim WindowName As String
Dim WorkbooksEntry1 As String
Dim WorkbooksEntry2 As String
'(*PiReport
Dim VP_iter_Pi As Integer
Dim VP_Start_Pi As Integer
Dim TotalVPs_Pi As Integer
Dim PiStyle As Long
Dim PiStyleDelta As Long
Dim secMax_Activity As Integer
')*Variable declaration

'(*Preparations before running the macro
'Manual tasks/settings before starting the macro
'1. With IE 7 make sure you have only one tab open (the one with SmC)!!!
'2. Module>Projects Style:Main R&D PI Export_3
'3. Module>Projects>Open Style: R&D Reporting Master Data Set_5
'4. Module>Projects>Open Scheduling>Hours&expenditures>One line per resource
'5. choose only Pi download yes=1 (PiReport) or no=0 (ResourceReport)
'default value is 0 (ResourceReport),
'if set to 1, then choose the SmC virtual portfolio just before the first PI virtual portfolio
'if you set BySyngentaPortfolio=True then you have 15 seconds to set the right VP
'PlandActivity = 11 'default=0,
'6. choose Smc user rights with and without read only
delta1 = 0 'read only: 0, all rights: 17

```

```

    delta2 = 0 'read only: 0, all rights: 56
'7. Calibrate the mouse click on the Open-Module button
    Target_X0 = 21 '(calibrate here)
    Target_Y0 = 137 '(calibrate here)
'8. set the path and the name of the workbook that is created (or opened/activated if it already
exists)
    WbPath = "C:\Users\t740698\Desktop\"
    wbdate = z_wbdate(, Now())
    WbName = "CONFIDENTIAL_SmC_Download" & "_" & wbdate & "_V1-0" & ".xlsb"
    'wbname = "SmC_Download.xlsb"
'9. Click on the BySyngentaPortfolio-Tab
    'default value is true (Click)
    'if set to false, then make sure that the tab in SmC is set to BySyngentaPortfolio
    BySyngentaPortfolio = True 'NoClick=False,Click=True
'10.Set the Reset-Button for the Resource report
    'default value is 1
    'only used in combination with PlandActivity=0 (ResourceReport), no reset with PlandActivity=1
(PiReport)
    ResetButtonYes = 1 'Yes=1, No=0 (does not influence PiReport)
'11.Resource Report: Number of virtual portfolios
    TotalVPs = 65 '65 if there are 66 (because it starts with 0)
'12.Resource Style
    'default=0 (manually set under point 3, no clicking)
    ResourceStyle = 0
    ResourceStyleDelta = 300 'determine the right value
'13.Max waiting time for the download IE Window
    'default=35 seconds
    secMax_Activity = 120 '60 '38
    secMax_PI = 220
'14.After a new SmC release check all positions of buttons that are clicked by the program
    'find: (new position and mouse click)
'15.Check from time to time whether the waiting times before new clicks are enough
'16.Pi Style
    'default=0 (manually set under point 2, no clicking)
    PiStyle = 0
    PiStyleDelta = 300 'determine the right value
'17.PiReport: Number of virtual portfolios
    TotalVPs_Pi = 11 '11 if there are 12 Pi virtual portfolios(because it starts with 0)
'18.IE Zoom must be set to 100% (IE cell in the lower left corner)
'19.Selection problem workaround
    'number of clicks on the scroll bar to have all PIs selected
    ScrollClicks_Iter_Max_Big = 9
    ScrollClicks_Iter_Max_Medium = 7
    ScrollClicks_Iter_Max_Small = 5
    ScrollClicks_Iter_Max_VerySmall = 3
'20.Check that in the one line per resource report all Attributes are visible
'21.Set the resource report
    'none=0, OneLinePerResource=1, Gant=2
    'default=2 since OneLinePerResource sets back the borderline between table and graph
    'ResourceReportFlag = 2
'22.Make sure Outlook Meeting request reminder does not pop up

')*Preparations before running the macro

```

```

'(*Create a new Excel Workbook
'show the desktop (minimize all windows)
Call z_ShowDesktop
Call z_Wait2(100)
'add, open or activate an Excel workbook (and session)
Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, mywb)
Call z_Wait2(3000)
'move and resize excel session
Call z_ExcelSessionWindowNormal(mywb)
Call z_ExcelSessionWindowMoveAndResize(mywb, "400", "0", "700", "340")
xlapp.Wait (Now + TimeValue("0:00:03"))
'minimize all Excel workbooks within the Excel session except mywb
Call z_ExcelWorkbookWindowMinimizeAll(mywb)
Call z_ExcelWorkbookWindowNormal(mywb)
'do some renaming, deleting and saving
'open or activate mywb
Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, mywb)
'bring the Excel session into xlnormal
Call z_ExcelSessionWindowNormal(mywb)
Call z_ExcelWorkbookWindowMaximized(mywb)
'do the next steps only if they are not already done
On Error Resume Next
'rename logfile
mywb.Sheets("Sheet1").Select
Sh_log = "Logfile"
mywb.Sheets("Sheet1").name = Sh_log
'delete empty sheets
Call z_DeleteWbSheet(mywb, "Sheet2")
Call z_DeleteWbSheet(mywb, "Sheet3")
On Error GoTo 0
'save workbook
mywb.Save
'turn off the alerts!!
mywb.Application.DisplayAlerts = False
')*Create a new Excel Workbook

'(*prepare the SmC window in an IE object
'With IE 7 make sure you have only one tab open (the one with SmC)!!!
'or find a solution to toggle between the tabs (send key Ctrl+Tab) until you found the SmC
'IE object instantiation
'myPageURL = "smartchoice.pro.intra" 'myPageURL = "smartchoice.stg.intra"
'Only if not already Open IE: Open IE, Resize, Reposition, Start SmC, Wait 30secs
myPageTitle = "SmartChoice"
Set myIE = IE_Preparation(myPageTitle, myPageURL)
Call z_Wait2(100)
'get the IE handle
My_IE_hWnd = myIE.hwnd
'show or restore IE depending on its current state (iconic = minimized)
If IsIconic(My_IE_hWnd) Then
    Call ShowWindow(My_IE_hWnd, SW_RESTORE)
Else
    Call ShowWindow(My_IE_hWnd, SW_SHOW)

```

```

End If
'bring SmC IE to the foreground
SetForegroundWindow My_IE_hWnd
')*prepare the SmC window in an IE object

If PlandActivity = "11" Or PlandActivity = "01" Then
    Call z_ExcelSessionWindowMinimized(mywb)
    Call z_ExcelSessionWindowNormal(mywb)
    notused = InputBox("set the export to 1997-2000", , "no entry", 13500, 12500)
    If notused <> "no entry" Then
        Stop
        Call z_Wait2(2000)
    End If
    Call z_Wait2(2000)
    Call SetForegroundWindow(My_IE_hWnd)
End If

'(*Open the SmC Project-Module
    Call z_Wait2(2000)
'Open-Module-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
    Call z_Wait2(100)
'Project-Module(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
    Call z_Wait2(15000)
'BySyngentaPortfolio-Tab(new position and mouse click)
If BySyngentaPortfolio Then
    'new position and mouse click
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 144, 123)
    Call z_Wait2(15000)
Else
    Call z_Wait2(100)
End If
')*Open the SmC Project-Module

'(*ResetButton
If PlandActivity = "11" Or PlandActivity = "01" Then
    If ResetButtonYes Then
        'VirtualPortfolioAdmin-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 238 + delta1, 63)
        Call z_Wait2(100)
        'Reset-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 238 + delta1, 148)
        Call z_Wait2(2000)
    Else
        Call z_Wait2(100)
    End If
Else
    'Select the last activity virtual portfolio, so that the programm starts with
    'selecting the next virtual portfolio which is the first Pi virtual portfolio
    End If
')*ResetButton

```

```

If PlandActivity = "11" Or PlandActivity = "01" Then
    Call z_ExcelSessionWindowMinimized(mywb)
    Call z_ExcelSessionWindowNormal(mywb)
    notused = InputBox("set the style to: return to default style on PI report (faster selection)", , "no
entry", 13500, 12500)
    If notused <> "no entry" Then
        Stop
        Call z_Wait2(2000)
    End If
    Call z_Wait2(2000)
    Call SetForegroundWindow(My_IE_hWnd)
End If

```

```

'(*Resource Report
Call z_Wait2(2000)
If PlandActivity = "11" Or PlandActivity = "01" Then
    FailedDownloads_iter = 1
    VP_Start = 0 '0
    'loop over the resource VPs
    For VP_iter = VP_Start To TotalVPs

        '**Bring IE to the foreground
        'show or restore IE depending on its current state
        If IsIconic(My_IE_hWnd) Then
            Call ShowWindow(My_IE_hWnd, SW_RESTORE)
        Else
            Call ShowWindow(My_IE_hWnd, SW_SHOW)
        End If
        'bring SmC IE to the foreground
        Call SetForegroundWindow(My_IE_hWnd)
        'To be sure SmC is in an idle state before going back to the portfolio module
        Call z_Wait2(3000)
    '**Bring IE to the foreground

    '**close the activities or not and bring SmC IE back to the modul "project"
    If VP_iter <> 0 Then
        'Open-Module-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
        Call z_Wait2(3000)
        'Project-Module(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
        Call z_Wait2(5000) '(15000)
    End If
    '**close the activities or not and bring SmC IE back to the modul "project"

    '**Choose a new Virtual Portfolio
    'VirtualPortfolioChoice-DropDown(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 533 + delta1, 63)
    Call z_Wait2(5000)
    'VirtualPortfolioChoice(new position and mouse click)
    VP_DropdownSecondPos = 97 'Delta Y0 to click on the second VP
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 338 + delta1,
VP_DropdownSecondPos)

```



```

    Call z_Wait2(8000) '(15000)
  '**Choose a new Virtual Portfolio

  '**Workaround for the selection problem: clicks on the scroll bar
  '
  '0 means VP 0+1
  '1 means VP 1+1
  '2 means VP 2+1
  '
  'Nof PIs from 140-200 as well as the last one: VP_Iter = 58 (move it to the right place if it
  becomes the second before last)
  '
  'If VP_iter = 19 - 1 Or _
  '  VP_iter = 34 - 1 Or _
  '  VP_iter = 36 - 1 Or _
  '  VP_iter = 39 - 1 Or _
  '  VP_iter = 54 - 1 Or _
  '  VP_iter = 63 - 1 Then
  '    'SelectAll-Button(new position and mouse click)
  '    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
  '    call z_Wait2(100)
  '    For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Big
  '      'Scroll-Bar(new position and mouse click)
  '      '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
  disappears
  '        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
  '        call z_Wait2(2000)
  '        'SelectAll-Button(new position and mouse click)
  '        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
  '        call z_Wait2(100)
  '      Next
  '    'Nof PIs from 100-139
  '    Elseif VP_iter = 4 - 1 Or _
  '      VP_iter = 18 - 1 Or _
  '      VP_iter = 20 - 1 Or _
  '      VP_iter = 21 - 1 Or _
  '      VP_iter = 37 - 1 Or _
  '      VP_iter = 38 - 1 Or _
  '      VP_iter = 40 - 1 Or _
  '      VP_iter = 42 - 1 Or _
  '      VP_iter = 43 - 1 Or _
  '      VP_iter = 44 - 1 Or _
  '      VP_iter = 45 - 1 Or _
  '      VP_iter = 47 - 1 Or _
  '      VP_iter = 49 - 1 Or _
  '      VP_iter = 53 - 1 Or _
  '      VP_iter = 56 - 1 Or _
  '      VP_iter = 58 - 1 Or _
  '      VP_iter = 61 - 1 Or _
  '      VP_iter = 62 - 1 Then
  '        'SelectAll-Button(new position and mouse click)
  '        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
  '        call z_Wait2(100)
  '        For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Medium
  '          'Scroll-Bar(new position and mouse click)

```

```

'       '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'       call z_Wait2(2000)
'       'SelectAll-Button(new position and mouse click)
'       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'       call z_Wait2(100)
'       Next
'       'Nof Pls from 60-99
'       Elseif VP_iter = 2 - 1 Or _
'       VP_iter = 3 - 1 Or _
'       VP_iter = 6 - 1 Or _
'       VP_iter = 22 - 1 Or _
'       VP_iter = 23 - 1 Or _
'       VP_iter = 29 - 1 Or _
'       VP_iter = 32 - 1 Or _
'       VP_iter = 33 - 1 Or _
'       VP_iter = 41 - 1 Or _
'       VP_iter = 46 - 1 Or _
'       VP_iter = 48 - 1 Or _
'       VP_iter = 50 - 1 Or _
'       VP_iter = 51 - 1 Or _
'       VP_iter = 55 - 1 Or _
'       VP_iter = 59 - 1 Or _
'       VP_iter = 60 - 1 Then
'       'SelectAll-Button(new position and mouse click)
'       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'       call z_Wait2(100)
'       For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_Small
'       'Scroll-Bar(new position and mouse click)
'       '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'       call z_Wait2(2000)
'       'SelectAll-Button(new position and mouse click)
'       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'       call z_Wait2(100)
'       Next
'       'Nof Pls from 1-59
'       Else
'       For ScrollClicks_Iter = 1 To ScrollClicks_Iter_Max_VerySmall
'       'Scroll-Bar(new position and mouse click)
'       '707 instead of 691 if you use the PI default style because then the horizontal scrollbar
disappears
'       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 691)
'       call z_Wait2(2000)
'       'SelectAll-Button(new position and mouse click)
'       Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
'       call z_Wait2(100)
'       Next
'       End If

'SelectAll-Button(new position and mouse click)

```

```

Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
Call z_Wait2(100)
'Click onto the scroll bar
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 707)
Call z_Wait2(2000)
'SelectAll-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
Call z_Wait2(100)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1158, 707)
Call z_Wait2(2000)
'SelectAll-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
Call z_Wait2(100)
')**Workaround for the selection problem: clicks on the scroll bar

'(**Open the OneLinePerResource or Gant Report
'SelectAll-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
Call z_Wait2(5000)
'Open-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 120 + delta2, 64)
Call z_Wait2(100)
'OpenSelectedProjects-Button(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 131 + delta2, 91)
Call z_Wait2(8000) '(15000)
'Choose the one line per resource report or the gant report in the first loop
If VP_iter = 0 Then
    If ResourceReportFlag = 1 Then
        'Scheduling-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 29, 30)
        Call z_Wait2(3000)
        'Hours&Expenditures-DropDownEntry(new position no click!)
        Call z_SetMousePos(Target_X0, Target_Y0, 29, 163)
        Call z_Wait2(3000)
        'OneLinePerResource-DropDownEntry(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 239, 163)
        Call z_Wait2(3000)
    ElseIf ResourceReportFlag = 2 Then
        'Scheduling-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 29, 30)
        Call z_Wait2(3000)
        'Gant-DropDownEntry(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 29, 63)
        Call z_Wait2(3000)
    ElseIf ResourceReportFlag = 3 Then
        'Tracking-Button(new position and mouse click)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 180, 30)
        Call z_Wait2(3000)
        'ViewDetailedCosts-DropDownEntry(new position and mouse click)
        Call z_SetMousePos(Target_X0, Target_Y0, 180, 85)
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 350, 85)
        Call z_Wait2(10000)
    End If

```

```

End If
'select the "R&D Reporting Master Data Set" style
If ResourceStyle Then
    'Style-Button(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 676, 135)
    Call z_Wait2(3000)
    'Style-Choice(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 580, ResourceStyleDelta)
    Call z_Wait2(15000)
End If
')**Open the OneLinePerResource or Gant Report

If VP_iter = 0 Then
    Call z_ExcelSessionWindowMinimized(mywb)
    Call z_ExcelSessionWindowNormal(mywb)
    If ResourceReportFlag = 2 Then
        notused = InputBox("set the style to: R&D Reporting Master DataSet_6 for the Activity
download", , "no entry", 13500, 12500)
    ElseIf ResourceReportFlag = 3 Then
        notused = InputBox("set the style to: BC Report; Unit=SalesIncrem.&SalesCanibal.", , "no
entry", 13500, 12500)
    End If
    If notused <> "no entry" Then
        Stop
        Call z_Wait2(2000)
    End If
    Call z_Wait2(2000)
    Call SetForegroundWindow(My_IE_hWnd)
End If

'(**Excel Download
ExcelDownload_1:
    ***Excel-Download-Button(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1025, 0)
    Call z_Wait2(1)
    ***Minimize SmC IE
    Call ShowWindow(My_IE_hWnd, SW_MINIMIZE)
    DoEvents
    Call z_Wait2(1)
    ***make excel active
    Call z_ExcelSessionWindowNormal(mywb)
    Call z_ExcelWorkbookWindowNormal(mywb)
    DoEvents
    Call z_Wait2(10)
    ***Determine when the Download IE appears
    'if the time secMax is not exceeded, the download should work out
    'if the time secMax is exceeded, the download is about to fail, wait an try to close the IE
window
    'that appears after secMax. After some time this window gets the name HTTP 500 ...
    strWindowTitle = "" & "opxscp.eame.syngenta" & ""
    sec = z_IE_WaitUntilNewWindowExists_HTML_Export(strWindowTitle, secMax_Activity)
    DoEvents
    ***If IE download fails (HTTP 500 ...), wait and close IE

```

```

Dim flag_1 As Boolean
flag_1 = False
Do Until flag_1 = True 'endless loop
    Dim ExcelDownload_Succeeded As Boolean
    ExcelDownload_Succeeded = False
    ExcelDownload_Succeeded = z_ExcelDownloadExists(mywb)
    If ExcelDownload_Succeeded = True Then
        Exit Do
    End If
    strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet
Explorer provided by Syngenta"
    Dim ExcelDownload_Failed As Boolean
    ExcelDownload_Failed = False
    ExcelDownload_Failed = z_WindowExists(strWindowTitle)
    If ExcelDownload_Failed = True Then
        Exit Do
    End If
Loop
DoEvents
'    'to be sure SmC is in an idle state before going back to the portfolio module
'    If sec > secMax_Activity Or sec = secMax_Activity Then
'        'wait until SmC is in an idle state
'        Call z_Wait2(100000)
'        'try to close the IE window
'        strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided
by Syngenta"
'        Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
'        End If
'    ***minimize IE that prepares the download and wait until the IE download is exported into
Excel
    strWindowTitle = "*" & "opxscp.eame.syngenta" & "*"
    My_IE_XL_hWnd = z_FindWindowHandle(strWindowTitle)
    If My_IE_XL_hWnd <> 0 Then
        'minimize IE
        Call ShowWindow(My_IE_XL_hWnd, SW_MINIMIZE)
        'give some waiting time until the IE download is exported into Excel
        ' call z_Wait2(10000)
        'strWindowTitle = "Microsoft Excel"
        'Dim XL_hWnd As Long
        'XL_hWnd = z_FindWindowHandle(strWindowTitle)
        'SetForegroundWindow XL_hWnd
        'XL_hWnd_ = mywb.Application.hwnd
        'SetForegroundWindow XL_hWnd_
        'SendKeys "{TAB}"
        'SendKeys "{TAB}"
        'SendKeys "{Enter}"
        'DoEvents
        'mywb.Application.DisplayAlerts = True
    End If
    ***write out the window name
    WindowName = Workbooks(Workbooks.Count).name
    WorkbooksEntry1 = Workbooks(Workbooks.Count).Sheets(1).Cells(2, 5)
    WorkbooksEntry2 = Workbooks(Workbooks.Count).Sheets(1).Cells(2, 6)

```

```

mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 5) = VP_iter + 1
mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 6) = sec
mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 7) = WindowName
mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 8) = WorkbooksEntry1
mywb.Worksheets(Sh_log).Cells(VP_iter + 2, 9) = WorkbooksEntry2
***Move the newly created download Wb into the SmC_Download file
Call z_MoveWbSheetsIntoAnotherWb_V2(mywb, "SmC_A_VP_", VP_iter + 1)
Call z_Wait2(100)
***minimize all Excel workbooks within the Excel session
Call z_ExcelWorkbookWindowMinimizeAll(mywb)
DoEvents
Call z_Wait2(100)
*** If IE fails, or IE failed to close before, so try to close the window now
strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet
Explorer provided by Syngenta"
Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
Call z_Wait2(100)
strWindowTitle = "Verify - Windows Internet Explorer provided by Syngenta"
Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
Call z_Wait2(100)
strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided by
Syngenta"
Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter, FailedDownloads_iter)
Call z_Wait2(100)

If ExcelDownload_Failed = True Then
  If ExcelDownload_Succeeded = False Then
    Call ShowWindow(My_IE_hWnd, SW_MAXIMIZE)
    DoEvents
    Call z_Wait2(100)
    GoTo ExcelDownload_1:
  End If
End If

')**Excel Download
Next VP_iter
'save workbook
mywb.Save
DoEvents
Call z_Wait2(10000)
End If
')*Resource Report

'(*PiReport
'(**close the activities or not and bring SmC IE back to the modul "project"
If PlandActivity = "11" Or PlandActivity = "01" Then
  'Open-Module-Button(new position and mouse click)
  Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
  Call z_Wait2(3000)
  'Project-Module(new position and mouse click)
  Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
  Call z_Wait2(14000)
End If

```

```

')**close the activities or not and bring SmC IE back to the modul "project"

If PlandActivity = "11" Or PlandActivity = "01" Or PlandActivity = "10" Then
    Call z_ExcelSessionWindowMinimized(mywb)
    Call z_ExcelSessionWindowNormal(mywb)
    notused = InputBox("set the export to 2007-2010 for the PI download", , "no entry", 13500,
12500)
    If notused <> "no entry" Then
        Stop
        Call z_Wait2(2000)
    End If
    notused = InputBox("set the style to Main R&D PI Export_6 for the PI download", , "no entry",
13500, 12500)
    If notused <> "no entry" Then
        Stop
        Call z_Wait2(2000)
    End If
    notused = InputBox("set the VP before the first PI VP", , "no entry", 13500, 12500)
    If notused <> "no entry" Then
        Stop
        Call z_Wait2(2000)
    End If
    Call z_Wait2(2000)
    Call SetForegroundWindow(My_IE_hWnd)
End If

If PlandActivity = "11" Or PlandActivity = "01" Or PlandActivity = "10" Then
    'loop over all Pi virtual portfolios
    VP_Start_Pi = 0
    For VP_iter_Pi = VP_Start_Pi To TotalVPs_Pi

        '(**Bring IE to the foreground
        'show or restore IE depending on its current state
        If IsIconic(My_IE_hWnd) Then
            Call ShowWindow(My_IE_hWnd, SW_RESTORE)
        Else
            Call ShowWindow(My_IE_hWnd, SW_SHOW)
        End If
        'bring SmC IE to the foreground
        Call SetForegroundWindow(My_IE_hWnd)
        'To be sure SmC is in an idle state before going back to the portfolio module
        Call z_Wait2(5000)
    ')**Bring IE to the foreground

    '(**close the activities or not and bring SmC IE back to the modul "project"
    '    If VP_iter_Pi = VP_Start_Pi Then
    '        'Open-Module-Button(new position and mouse click)
    '        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
    '        call z_Wait2(100))
    '        'Project-Module(new position and mouse click)
    '        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
    '        Call z_Wait2(14000)
    '    End If

```

```

'    '**close the activities or not and bring SmC IE back to the modul "project"

'(**select the "Main R&D PI Report" style
If PiStyle Then
    'Style-Button(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1157, 141)
    Call z_Wait2(3000)
    'Style-Choice(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1096, PiStyleDelta)
    Call z_Wait2(10000)
End If
'(**select the "Main R&D PI Report" style

'(**Choose a new Virtual Portfolio
'VirtualPortfolioChoice-DropDown(new position and mouse click)
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 533 + delta1, 63)
Call z_Wait2(5000)
'VirtualPortfolioChoice(new position and mouse click)
VP_DropdownSecondPos = 97 'Delta Y0 to click on the second VP
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 338 + delta1,
VP_DropdownSecondPos)
Call z_Wait2(15000)
'(**Choose a new Virtual Portfolio

'(**Excel Download
ExcelDownload_Failed = False
ExcelDownload_2:
    '**Excel-Download-Button(new position and mouse click)
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1025, 0)
    Call z_Wait2(2000)
    '**Minimize SmC IE
    Call ShowWindow(My_IE_hWnd, SW_MINIMIZE)
    DoEvents
    Call z_Wait2(100)
    '**make excel active
    Call z_ExcelSessionWindowNormal(mywb)
    Call z_ExcelWorkbookWindowNormal(mywb)
    DoEvents
    Call z_Wait2(100)
    '**Determine when the Download IE appears
    'if the time secMax is not exceeded, the download should work out
    'if the time secMax is exceeded, the download is about to fail, wait an try to close the IE
window
    'that appears after secMax. After some time this window gets the name HTTP 500 ...
    DurationSinceDownloadClick = 4
    strWindowTitle = "" & "opxscp.eame.syngenta" & ""
    sec = z_IE_WaitUntilNewWindowExists_HTML_Export(strWindowTitle, secMax_PI -
DurationSinceDownloadClick)
    sec = sec + DurationSinceDownloadClick
    DoEvents

    '**If IE download fails (HTTP 500 ...), wait and close IE
    flag_1 = False

```



```

Do Until flag_1 = True 'endless loop

    ExcelDownload_Succeeded = False
    ExcelDownload_Succeeded = z_ExcelDownloadExists(mywb)
    If ExcelDownload_Succeeded = True Then
        Exit Do
    End If
    strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet
Explorer provided by Syngenta"

    ExcelDownload_Failed = False
    ExcelDownload_Failed = z_WindowExists(strWindowTitle)
    If ExcelDownload_Failed = True Then
        Exit Do
    End If
Loop
DoEvents

'to be sure SmC is in an idle state before going back to the portfolio module
'    If sec > secMax_PI Or sec = secMax_PI Then
'        'wait until SmC is in an idle state
'        Call z_Wait2(100000) 'maybe enhance to 10 minutes
'        'try to close the IE window
'        strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided
by Syngenta"
'        Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter_Pi, FailedDownloads_iter)
'    End If
'*** minimize IE that prepares the download and wait until the IE download is exported into
Excel
strWindowTitle = "" & "opxscp.eame.syngenta" & ""
My_IE_XL_hWnd = z_FindWindowHandle(strWindowTitle)
If My_IE_XL_hWnd <> 0 Then
    'minimize IE
    Call ShowWindow(My_IE_XL_hWnd, SW_MINIMIZE)
    'give some waiting time until the IE download is exported into Excel
    DoEvents
    Call z_Wait2(10000)
End If
'*** write out the window name
WindowName = Workbooks(Workbooks.Count).name
WorkbooksEntry1 = Workbooks(Workbooks.Count).Sheets(1).Cells(2, 5)
WorkbooksEntry2 = Workbooks(Workbooks.Count).Sheets(1).Cells(2, 6)
mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 5) = VP_iter_Pi + 1
mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 6) = sec
mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 7) = WindowName
mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 8) = WorkbooksEntry1
mywb.Worksheets(Sh_log).Cells(VP_iter_Pi + 2, 9) = WorkbooksEntry2
'*** Save the export workbook
xlapp.DisplayAlerts = False
Call z_WorkbookSave(ActiveWorkbook)
xlapp.DisplayAlerts = True
'*** Move newly created download Wb into the SmC_Download file
Call z_MoveWbSheetsIntoAnotherWb_V2(mywb, "SmC_P_VP_", VP_iter_Pi + 1)

```

```

        Call z_Wait2(100)
    *** Save the download workbook
        Call z_WorkbookSave(mywb)
    *** minimize all Excel workbooks within the Excel session
        Call z_ExcelWorkbookWindowMinimizeAll(mywb)
        DoEvents
        Call z_Wait2(100)
    *** If IE fails, or IE failed to close before, so try to close the window now
        strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet
Explorer provided by Syngenta"
        Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter_Pi, FailedDownloads_iter)
        Call z_Wait2(100)
        strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided by
Syngenta"
        Call z_CloseIE3(strWindowTitle, Sh_log, VP_iter_Pi, FailedDownloads_iter)
        Call z_Wait2(100)
        If ExcelDownload_Failed = True Then
            If ExcelDownload_Succeeded = False Then
                Call ShowWindow(My_IE_hWnd, SW_MAXIMIZE)
                DoEvents
                Call z_Wait2(100)
                GoTo ExcelDownload_2:
            End If
        End If
    End If
    '(**Excel Download
    Next VP_iter_Pi
    'save workbook
    mywb.Save
End If
'(*PiReport
notused = InputBox("Download has been finished", , "no entry", 13500, 12500)
'Call z_TrackTime("C:\Users\t740698\Desktop\TimeTracking.txt", _
    Array("Main_DownloadReport", "End", "task: ", CStr(Now()), ""))

End Sub

```

```

Function z_wbdate(Optional sDt As String, Optional Dt As Date) As String
    Dim Yr As String
    Dim Mt As String
    Dim Dy As String
    Dim sDt_arr As Variant
    If sDt <> Empty Then
        sDt_arr = Split(sDt, "_")
        Yr = sDt_arr(0)
        Mt = sDt_arr(1)
        Dy = sDt_arr(2)
    Else
        'If Not Dt Is Nothing Then
        If Dt <> Empty Then
            Yr = VBA.DatePart("Y", Dt)

```

```

        Mt = VBA.DateTime.Month(Dt)
        Dy = VBA.DateTime.Day(Dt)
    Else
        Yr = VBA.DateTime.Year(Now())
        Mt = VBA.DateTime.Month(Now())
        Dy = VBA.DateTime.Day(Now())
    End If
End If

If Mt < 10 Then
    Mt = "0" & Mt
End If
If Dy < 10 Then
    Dy = "0" & Dy
End If
z_wbdate = Yr & "_" & Mt & "_" & Dy
End Function

```

Function z\_WorkbookNewOrOpenOrActivate(WbName As String, WbPath As String, ByRef Wb As Workbook)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 19.10.2011

'Objective: Open the Workbook RD.xlsb if not already open and activate it.

'look if wbName is existent in workbooks list

Dim i As Long

For i = Workbooks.Count To 1 Step -1

    If Workbooks(i).name = WbName Then Exit For

Next

'if wbName is existent in workbooks list, then i<>0-> activate workbook

'if wbName is not existent then i=0-> open workbook, activate workbook

If i <> 0 Then

    Set Wb = VBA.Interaction.GetObject(WbPath & WbName)

    Wb.Activate

Else

    On Error GoTo NewWB

    Set Wb = Workbooks.Open(WbPath & WbName)

    Wb.Activate

    On Error GoTo 0

End If

Exit Function

NewWB:

Set Wb = Workbooks.Add

Dim FileFormatValue As Integer

If WbName <> Empty Then

    Select Case LCase(Right(WbName, Len(WbName) - InStrRev(WbName, ".", , 1)))

        Case "xls": FileFormatValue = 56

        Case "xlsx": FileFormatValue = 51

        Case "xlsm": FileFormatValue = 52

        Case "xlsb": FileFormatValue = 50

        Case Else: FileFormatValue = 0

    End Select

End If

```

    Wb.SaveAs FileName:=WbPath & WbName, FileFormat:=FileFormatValue
End Function
Function z_ExcelSessionWindowNormal(Optional ByRef Wb As Workbook)
    'Fenster der Excel session
    If Wb Is Nothing Then
        Application.WindowState = xlNormal
    Else
        Wb.Application.WindowState = xlNormal
        Wb.Activate
    End If
End Function

Function z_ExcelSessionWindowMinimized(Optional ByRef Wb As Workbook)
    'Fenster der Excel session
    If Wb Is Nothing Then
        Application.WindowState = xlMinimized
    Else
        Wb.Application.WindowState = xlMinimized
    End If
End Function

Function z_ExcelSessionWindowMoveAndResize(Optional ByRef Wb As Workbook, _
    Optional top As Variant, Optional left As Variant, _
    Optional width As Variant, Optional height As Variant)
    On Error Resume Next
    If Wb Is Nothing Then
        If top <> Empty Then
            Application.top = top
        End If
        If left <> Empty Then
            Application.left = left
        End If
        If width <> Empty Then
            Application.width = width
        End If
        If height <> Empty Then
            Application.height = height
        End If
    Else
        If top <> Empty Then
            Wb.Application.top = CInt(top)
        End If
        If left <> Empty Then
            Wb.Application.left = CInt(left)
        End If
        If width <> Empty Then
            Wb.Application.width = CInt(width)
        End If
        If height <> Empty Then
            Wb.Application.height = CInt(height)
        End If
    End If
    On Error GoTo 0
End Function

```

```
Function z_ExcelWorkbookWindowMinimized(ByRef Wb As Workbook)
```

```
'Fenster innerhalb der Excel session (workbooks)
```

```
If Windows(Wb.name).Visible Then
```

```
    Wb.Application.ActiveWindow.WindowState = xlMinimized
```

```
End If
```

```
End Function
```

```
Function z_ExcelWorkbookWindowMaximized(ByRef Wb As Workbook)
```

```
'Fenster innerhalb der Excel session (workbooks)
```

```
If Wb.Application.Windows(Wb.name).Visible Then
```

```
    Wb.Application.ActiveWindow.WindowState = xlMaximized
```

```
End If
```

```
End Function
```

```
Function z_ExcelWorkbookWindowNormal(ByRef Wb As Workbook)
```

```
'Fenster innerhalb der Excel session (workbooks)
```

```
If Wb.Application.Windows(Wb.name).Visible Then
```

```
    Wb.Application.ActiveWindow.WindowState = xlMaximized
```

```
End If
```

```
End Function
```

```
Function z_ExcelWorkbookWindowMinimizeAll(ByRef Wb_ref As Workbook)
```

```
    Application.ScreenUpdating = False
```

```
    Dim Wb As Workbook
```

```
    For Each Wb In Wb_ref.Application.Workbooks ' Workbooks
```

```
        'only those with status visible
```

```
        If Wb.Application.Windows(Wb.name).Visible Then
```

```
            Wb.Application.ActiveWindow.WindowState = xlMinimized
```

```
        End If
```

```
    Next
```

```
    Application.ScreenUpdating = True
```

```
End Function
```

```
Function z_DeleteWbSheet(Wb As Workbook, Sh As String)
```

```
    Application.DisplayAlerts = False
```

```
    Wb.Activate
```

```
    Dim WSh As Excel.Worksheet
```

```
    Set WSh = Sheets(Sh)
```

```
    WSh.Select
```

```
    Wb.Application.ActiveWindow.SelectedSheets.Delete
```

```
    Application.DisplayAlerts = True
```

```
End Function
```

```
Function z_WorkbookSaveAs(Wb_New_Path As String, Wb_New_Name As String, Optional wb_old
```

```
As Workbook, Optional wb_new As Workbook)
```

```
    If Not wb_old Is Nothing Then
```

```
        wb_old.Activate
```

```
    Else
```

```
        ActiveWorkbook.Activate
```

```
    End If
```

```
    'ChDir Wb_New_Path
```

```
    'ActiveWorkbook.SaveAs Filename:= _
```

```
    ' Wb_New_Path & Wb_New_Name _
```

```
    ' , FileFormat:=xlExcel12, CreateBackup:=False
```

```
    If Wb_New_Name <> Empty Then
```

```

        Select Case LCase(Right(Wb_New_Name, Len(Wb_New_Name) - InStrRev(Wb_New_Name, ".", ,
1)))
        Case "xls": FileFormatValue = 56
        Case "xlsx": FileFormatValue = 51
        Case "xlsm": FileFormatValue = 52
        Case "xlsb": FileFormatValue = 50
        Case Else: FileFormatValue = 0
    End Select
End If
'close wb with same name if open and then overwrite it
Application.DisplayAlerts = False
On Error GoTo ExistsAndOpen:
ActiveWorkbook.SaveAs FileName:=Wb_New_Path & Wb_New_Name,
FileFormat:=FileFormatValue
On Error GoTo 0
Application.DisplayAlerts = True
Set wb_new = ActiveWorkbook
Set wb_old = Nothing 'Wb_old was closed after saving and Wb_old was assigned the newly created
file
Exit Function
ExistsAndOpen:
    Workbooks(Wb_New_Name).Close False
    Resume
End Function
Function z_WorkbookSave(Wb As Workbook)
    Wb.Save
End Function

'-----Wait
Option Explicit

Private Type FILETIME
    dwLowDateTime As Long
    dwHighDateTime As Long
End Type

Private Const WAIT_OBJECT_0& = 0

Private Const INFINITE = &HFFFF
Private Const ERROR_ALREADY_EXISTS = 183&

Private Const QS_HOTKEY& = &H80
Private Const QS_KEY& = &H1
Private Const QS_MOUSEBUTTON& = &H4
Private Const QS_MOUSEMOVE& = &H2
Private Const QS_PAINT& = &H20
Private Const QS_POSTMESSAGE& = &H8
Private Const QS_SENDMESSAGE& = &H40
Private Const QS_TIMER& = &H10
Private Const QS_MOUSE& = (QS_MOUSEMOVE Or QS_MOUSEBUTTON)
Private Const QS_INPUT& = (QS_MOUSE Or QS_KEY)
Private Const QS_ALLINPUT& = (QS_SENDMESSAGE Or QS_PAINT _

```

Or QS\_TIMER Or QS\_POSTMESSAGE Or QS\_MOUSEBUTTON \_  
Or QS\_MOUSEMOVE Or QS\_HOTKEY Or QS\_KEY)

Private Declare Function CreateWaitableTimer Lib "kernel32" \_  
Alias "CreateWaitableTimerA" (ByVal lpSemaphoreAttributes \_  
As Long, ByVal bManualReset As Long, ByVal lpName As String) \_  
As Long

Private Declare Function SetWaitableTimer Lib "kernel32" ( \_  
ByVal hTimer As Long, lpDueTime As FILETIME, ByVal lPeriod \_  
As Long, ByVal pfnCompletionRoutine As Long, \_  
ByVal lpArgToCompletionRoutine As Long, \_  
ByVal fResume As Long) As Long

Private Declare Function CloseHandle Lib "kernel32" (ByVal \_  
hObject As Long) As Long

Private Declare Function MsgWaitForMultipleObjects Lib \_  
"user32" (ByVal nCount As Long, pHandles As Long, ByVal \_  
fWaitAll As Long, ByVal dwMilliseconds As Long, ByVal \_  
dwWakeMask As Long) As Long

'-----Mouse

Declare Function SetCursorPos Lib "user32.dll" ( \_  
ByVal x As Long, \_  
ByVal y As Long) As Long

Declare Sub sleep Lib "kernel32.dll" Alias \_  
"Sleep" (ByVal dwMilliseconds As Long)

Declare Function GetCursorPos Lib "user32.dll" ( \_  
ByRef lpPoint As POINTAPI) As Long

Type POINTAPI

x As Long

y As Long

End Type

Declare Sub mouse\_event Lib "user32" \_  
(ByVal dwFlags As Long, ByVal dx As Long, \_  
ByVal dy As Long, ByVal cButtons As Long, \_  
ByVal dwExtraInfo As Long)

Public Const MOUSE\_LEFT = 0

Public Const MOUSE\_MIDDLE = 1

Public Const MOUSE\_RIGHT = 2

'-----keyboard

Private Declare Sub keybd\_event Lib "user32.dll" (ByVal bVk As Byte, ByVal bScan As Byte, ByVal  
dwFlags As Long, \_  
ByVal dwExtraInfo As Long)

Public Const VK\_STARTKEY = &H5B

```
Public Const VK_M = 77
Public Const KEYEVENTF_KEYUP = &H2
```

```
'-----Window
```

```
Declare Function PostMessage Lib "user32" _
    Alias "PostMessageA" _
    (ByVal hwnd As Long, _
    ByVal wParam As Long, _
    ByVal lParam As Long) As Long
```

```
Public Const WM_CLOSE = &H10
```

```
Declare Function MoveWindow Lib "user32.dll" ( _
    ByVal hwnd As Long, _
    ByVal x As Long, _
    ByVal y As Long, _
    ByVal nWidth As Long, _
    ByVal nHeight As Long, _
    ByVal bRepaint As Long) As Long
```

```
Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" ( _
    ByVal lpClassName As String, _
    ByVal lpWindowName As String) As Long
```

```
' Module Name: ModFindWindowLike
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)
' Written 02/06/2005
```

```
Declare Function EnumWindows Lib "user32" _
    (ByVal lpEnumFunc As Long, _
    ByVal lParam As Long) As Long
```

```
Declare Function GetWindowText Lib "user32" _
    Alias "GetWindowTextA" _
    (ByVal hwnd As Long, _
    ByVal lpString As String, _
    ByVal cch As Long) As Long
```

```
'Custom structure for passing in the parameters in/out of the hook enumeration function
```

```
'Could use global variables instead, but this is nicer.
```

```
Type FindWindowParameters
```

```
    strTitle As String 'INPUT
    hwnd As Long      'OUTPUT
```

```
End Type
```

```
' Module Name: Modz_SetForegroundWindow
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)
' Written 02/06/2005
```



```
Declare Function SetForegroundWindow Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Declare Function GetWindowThreadProcessId Lib "user32" _  
    (ByVal hwnd As Long, _  
    lpdwProcessId As Long) As Long
```

```
Declare Function IsIconic Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Declare Function ShowWindow Lib "user32" _  
    (ByVal hwnd As Long, _  
    ByVal nCmdShow As Long) As Long
```

```
Declare Function AttachThreadInput Lib "user32" _  
    (ByVal idAttach As Long, _  
    ByVal idAttachTo As Long, _  
    ByVal fAttach As Long) As Long
```

```
Declare Function GetForegroundWindow Lib "user32" _  
    () As Long
```

```
Public Const SW_RESTORE = 9
```

```
Public Const SW_SHOW = 5
```

```
' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel  
,
```

```
' sTitel: muss nicht der exakte Fenstertitel sein  
'     hier kann bspw. auch nur der Anfang des Fenstertitel  
'     angegeben werden, z.B.: Fenstertitel*  
,
```

```
' benötigte API-Deklarationen
```

```
Declare Function GetWindowTextLength Lib "user32" _  
    Alias "GetWindowTextLengthA" ( _  
    ByVal hwnd As Long) As Long
```

```
Declare Function GetWindow Lib "user32" ( _  
    ByVal hwnd As Long, _  
    ByVal wCmd As Long) As Long
```

```
Public Const GW_HWNDNEXT = 2
```

```
Declare Function FindWindowEx Lib "user32.dll" _  
    Alias "FindWindowExA" ( _  
    ByVal hwndParent As Long, _  
    ByVal hwndChildAfter As Long, _  
    ByVal lpszClass As String, _  
    ByVal lpszWindow As String) As Long
```

```

'-----Wait
Public Sub Wait(INumberOfSeconds As Long)
    Dim ft As FILETIME
    Dim IBusy As Long
    Dim IRet As Long
    Dim dblDelay As Double
    Dim dblDelayLow As Double
    Dim dblUnits As Double
    Dim hTimer As Long

    hTimer = CreateWaitableTimer(0, True, vbNullChar)

    If Err.LastDllError = ERROR_ALREADY_EXISTS Then
        '
    Else
        ft.dwLowDateTime = -1
        ft.dwHighDateTime = -1
        IRet = SetWaitableTimer(hTimer, ft, 0, 0, 0, 0)
    End If

    dblUnits = CDbI(&H10000) * CDbI(&H10000)
    dblDelay = CDbI(INumberOfSeconds) * 1000 * 10000

    ft.dwHighDateTime = -CLng(dblDelay / dblUnits) - 1
    dblDelayLow = -dblUnits * _
        (dblDelay / dblUnits - Fix(dblDelay / dblUnits))

    If dblDelayLow < CDbI(&H800000000) Then
        dblDelayLow = dblUnits + dblDelayLow
        ft.dwHighDateTime = ft.dwHighDateTime + 1
    End If

    ft.dwLowDateTime = CLng(dblDelayLow)
    IRet = SetWaitableTimer(hTimer, ft, 0, 0, 0, False)

    Do
        IBusy = MsgWaitForMultipleObjects(1, hTimer, False, _
            INFINITE, QS_ALLINPUT&)
        DoEvents
    Loop Until IBusy = WAIT_OBJECT_0

    CloseHandle hTimer
End Sub

'-----Mouse
'Die nachfolgende Prozedur simuliert den gewünschten Mausklick.
Public Sub z_SendMouseClicked(ByVal mButton As Long)
    Const MOUSEEVENTF_LEFTDOWN = &H2
    Const MOUSEEVENTF_LEFTUP = &H4
    Const MOUSEEVENTF_MIDDLEDOWN = &H20
    Const MOUSEEVENTF_MIDDLEUP = &H40
    Const MOUSEEVENTF_RIGHTDOWN = &H8

```

```
Const MOUSEEVENTF_RIGHTUP = &H10
```

```
If (mButton = MOUSE_LEFT) Then
    Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
    Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
ElseIf (mButton = MOUSE_MIDDLE) Then
    Call mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
    Call mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
Else
    Call mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
    Call mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
End If
End Sub
```

```
Sub z_SendMausDoubleClick(ByVal mButton As Long)
z_SendMouseClicked (mButton)
z_SendMouseClicked (mButton)
End Sub
```

```
Function z_SetMousePosAndLeftClick(Target_X0 As Long, Target_Y0 As Long, _
    Target_DeltaX As Long, Target_DeltaY As Long)
Dim Target_X As Long
Dim Target_Y As Long
Target_X = Target_X0 + Target_DeltaX
Target_Y = Target_Y0 + Target_DeltaY
SetCursorPos Target_X, Target_Y
Excel.Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
z_SendMouseClicked (MOUSE_LEFT)
Excel.Application.Wait (Now + TimeValue("0:00:01"))
End Function
```

```
Function z_SetMousePos(Target_X0 As Long, Target_Y0 As Long, _
    Target_DeltaX As Long, Target_DeltaY As Long)
Dim Target_X As Long
Dim Target_Y As Long
Target_X = Target_X0 + Target_DeltaX
Target_Y = Target_Y0 + Target_DeltaY
SetCursorPos Target_X, Target_Y
Excel.Application.Wait (Now + TimeValue("0:00:01"))
End Function
```

```
'-----keyobard
```

```
Function z_ShowDesktop()
```

```
'Keyboard: Windows button + M button shows the desktop
```

```
'Do not test with debug F8 -> VBA Windows hides!!!
```

```
'http://msdn.microsoft.com/en-us/library/ms646304\(VS.85\).aspx
```

```
'http://msdn.microsoft.com/en-us/library/dd375731\(v=VS.85\).aspx
```

```
'WinKey down
```

```
keybd_event VK_STARTKEY, 0, 0, 0
```

```
'M key down
```

```
keybd_event VK_M, 0, 0, 0
```

```

'M key up
keybd_event VK_M, 0, KEYEVENTF_KEYUP, 0
'WinKey up
keybd_event VK_STARTKEY, 0, KEYEVENTF_KEYUP, 0

'do not minimiz form itself
'Me.WindowState = vbMaximized
'Me.WindowState = vbNormal[/b]
End Function

'-----Window

Public Function z_FindWindowHandle(ByVal sTitle As String) As Long
' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel
,
' sTitel: muss nicht der exakte Fenstertitel sein
'     hier kann bspw. auch nur der Anfang des Fenstertitel
'     angegeben werden, z.B.: Fenstertitel*
,

Dim lngHwnd As Long
Dim sText As String
' Handel des ersten Fensters
lngHwnd = FindWindow(vbNullString, vbNullString)
' alle Fenster durchlaufen
Do While lngHwnd <> 0
' Fenstertitel ermitteln
sText = z_GetWindowTitle(lngHwnd)
'Debug.Print lngHwnd & " " & sText
If Len(sText) > 0 And LCase$(sText) Like LCase$(sTitle) Then
z_FindWindowHandle = lngHwnd: Exit Do
End If
' Handel des nächsten Fensters
lngHwnd = GetWindow(lngHwnd, GW_HWNDNEXT)
Loop
End Function

' Hilfsfunktion zum Ermitteln des Fenstertitels
Public Function z_GetWindowTitle(ByVal hwnd As Long) As String
Dim lResult As Long
Dim sTemp As String

lResult = GetWindowTextLength(hwnd) + 1
sTemp = Space(lResult)
lResult = GetWindowText(hwnd, sTemp, lResult)
z_GetWindowTitle = left(sTemp, Len(sTemp) - 1)
End Function

' Module Name: ModFindWindowLike
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)
' Written 02/06/2005
Public Function z_FindWindowLike(strWindowTitle As String) As Long

```

'We'll pass a custom structure in as the parameter to store our result...

Dim Parameters As FindWindowParameters

Parameters.strTitle = strWindowTitle ' Input parameter

Call EnumWindows(AddressOf EnumWindowProc, VarPtr(Parameters))

z\_FindWindowLike = Parameters.hwnd

End Function

Function EnumWindowProc(ByVal hwnd As Long, \_  
IParam As FindWindowParameters) As Long

Dim strWindowTitle As String

strWindowTitle = Space(260)

Call GetWindowText(hwnd, strWindowTitle, 260)

strWindowTitle = TrimNull(strWindowTitle) ' Remove extra null terminator

'Debug.Print strWindowTitle

If strWindowTitle Like IParam.strTitle Then

IParam.hwnd = hwnd 'Store the result for later.

EnumWindowProc = 0 'This will stop enumerating more windows

Else

EnumWindowProc = 1

End If

End Function

' Module Name: ModSetForegroundWindow

' (c) 2005 Wayne Phillips (<http://www.everythingaccess.com>)

' Written 02/06/2005

Function TrimNull(strNullTerminatedString As String)

Dim lngPos As Long

'Remove unnecessary null terminator

lngPos = InStr(strNullTerminatedString, Chr\$(0))

If lngPos Then

TrimNull = left\$(strNullTerminatedString, lngPos - 1)

Else

TrimNull = strNullTerminatedString

End If

End Function

Public Function z\_SetForegroundWindow(strWindowTitle As String) As Boolean

```

Dim MyAppHwnd As Long
Dim CurrentForegroundThreadID As Long
Dim NewForegroundThreadID As Long
Dim lngRetVal As Long
Dim blnSuccessful As Boolean
MyAppHwnd = z_FindWindowLike(strWindowTitle)
If MyAppHwnd <> 0 Then
    'We've found the application window by the caption
    CurrentForegroundThreadID = GetWindowThreadProcessId(GetForegroundWindow(), ByVal 0&)
0&)
    NewForegroundThreadID = GetWindowThreadProcessId(MyAppHwnd, ByVal 0&)
    'AttachThreadInput is used to ensure SetForegroundWindow will work
    'even if our application isn't currently the foreground window
    '(e.g. an automated app running in the background)
    Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, True)
    lngRetVal = SetForegroundWindow(MyAppHwnd)
    Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, False)
    If lngRetVal <> 0 Then
        'Now that the window is active, let's restore it from the taskbar
        If IsIconic(MyAppHwnd) Then
            Call ShowWindow(MyAppHwnd, SW_RESTORE)
        Else
            Call ShowWindow(MyAppHwnd, SW_SHOW)
        End If
        blnSuccessful = True
    Else
        MsgBox "Found the window, but failed to bring it to the foreground!"
    End If
Else
    'Failed to find the window caption
    'Therefore the app is probably closed.
    MsgBox "Application Window '" + strWindowTitle + "' not found!"
End If
z_SetForegroundWindow = blnSuccessful
End Function

Public Function z_SetForegroundWindow2(MyAppHwnd As Long) As Boolean
    Dim CurrentForegroundThreadID As Long
    Dim NewForegroundThreadID As Long
    Dim lngRetVal As Long
    Dim blnSuccessful As Boolean
    If MyAppHwnd <> 0 Then
        'We've found the application window by the caption
        CurrentForegroundThreadID = GetWindowThreadProcessId(GetForegroundWindow(), ByVal 0&)
        NewForegroundThreadID = GetWindowThreadProcessId(MyAppHwnd, ByVal 0&)
        'AttachThreadInput is used to ensure SetForegroundWindow will work
        'even if our application isn't currently the foreground window
        '(e.g. an automated app running in the background)
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, True)
        lngRetVal = SetForegroundWindow(MyAppHwnd)
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, False)
        If lngRetVal <> 0 Then
            'Now that the window is active, let's restore it from the taskbar
            If IsIconic(MyAppHwnd) Then

```

```

        Call ShowWindow(MyAppHWnd, SW_RESTORE)
    Else
        Call ShowWindow(MyAppHWnd, SW_SHOW)
    End If
    blnSuccessful = True
Else
    MsgBox "Found the window, but failed to bring it to the foreground!"
End If
Else
    'Failed to find the window caption
    'Therefore the app is probably closed.
    MsgBox "Application Window '" & " not found!"
End If
z_SetForegroundWindow2 = blnSuccessful
End Function

'-----Microsoft Internet Controls (Tools>References: Microsoft Internet Controls)
'returns new instance of Internet Explorer
Function GetNewIE() As SHDocVw.InternetExplorer
    'create new IE instance
    Set GetNewIE = New SHDocVw.InternetExplorer
    'start with a blank page
    GetNewIE.Navigate2 "about:Blank"
End Function

'loads a web page and returns True or False depending on
'whether the page could be loaded or not
Function LoadWebPage(i_IE As SHDocVw.InternetExplorer, _
    i_URL As String) As Boolean
    With i_IE
        'open page
        .Navigate i_URL
        'Window security
        If i_URL = "smartchoice.stg.intra" Then
            'here code that fills the pop up
        End If
        'wait until IE finished loading the page
        Do While .ReadyState <> READYSTATE_COMPLETE
            Excel.Application.Wait Now + TimeValue("0:00:01")
        Loop
        'check if page could be loaded
        '(does not work out if the page is relocated)
        'this is a workaround. Better: do check wheter a window "SmartChoice" exists!!
        If i_URL = "smartchoice.pro.intra" Or i_URL = "smartchoice.stg.intra" Then
            LoadWebPage = True
        Else
            If .Document.URL = i_URL Then
                LoadWebPage = True
            End If
        End If
    End With
End Function

```

```

Function z_IE_WaitUntilReadyStateComplete(i_IE As SHDocVw.InternetExplorer) As Integer
    Dim sec As Integer
    sec = 0
    'wait until IE finished loading the page
    Do While i_IE.ReadyState <> READYSTATE_COMPLETE
        Excel.Application.Wait Now + TimeValue("0:00:01")
        sec = sec + 1
    Loop
End Function

```

```

Function z_IE_WaitUntilNewWindowExists_XL_Export(strWindowTitle As String, secMax As Integer)
As Integer
    'function not used
    Dim sec As Integer
    sec = 0
    Dim MyAppHWND As Long
    Do While MyAppHWND = 0
        MyAppHWND = z_FindWindowLike(strWindowTitle)
        Call Wait(5) 'Excel.Application.Wait Now + TimeValue("0:00:01")
        DoEvents
        sec = sec + 1
        If sec = secMax Then
            z_IE_WaitUntilNewWindowExists_XL_Export = secMax
            Exit Function
        End If
    Loop
    'Do While MyAppHWND = 0
    '    MyAppHWND = z_FindWindowHandle(strWindowTitle)
    '    Excel.Application.Wait Now + TimeValue("0:00:01")
    '    sec = sec + 1
    '    If sec = secMax Then
    '        z_IE_WaitUntilNewWindowExists_XL_Export = secMax
    '        Exit Function
    '    End If
    'Loop
    z_IE_WaitUntilNewWindowExists_XL_Export = sec
End Function

```

```

Function z_IE_WaitUntilNewWindowExists_HTML_Export(strWindowTitle As String, secMax As
Integer) As Integer
    Dim sec As Integer
    sec = 0
    'Do While MyAppHWND = 0
    '    MyAppHWND = z_FindWindowLike(strWindowTitle)
    '    Excel.Application.Wait Now + TimeValue("0:00:01")
    '    sec = sec + 1
    '    If sec = secMax Then
    '        Exit Function
    '    End If
    'Loop
    Dim MyAppHWND As Long

```



```

Do While MyAppHWnd = 0
    MyAppHWnd = z_FindWindowHandle(strWindowTitle)
    Excel.Application.Wait Now + TimeValue("0:00:01")
    sec = sec + 1
    If sec = secMax Then
        z_IE_WaitUntilNewWindowExists_HTML_Export = secMax
        Exit Function
    End If
Loop
z_IE_WaitUntilNewWindowExists_HTML_Export = sec
End Function

```

```

Function z_WindowExists(strWindowTitle As String) As Boolean
    Dim MyAppHWnd As Long
    MyAppHWnd = z_FindWindowHandle(strWindowTitle)
    If MyAppHWnd <> 0 Then
        z_WindowExists = True
    Else
        z_WindowExists = False
    End If
End Function

```

```

'finds an open IE site by checking the URL
Function GetOpenIEByURL(ByVal i_URL As String) _
    As SHDocVw.InternetExplorer

```

```

Dim objShellWindows As New SHDocVw.ShellWindows

```

```

'ignore errors when accessing the document property
On Error Resume Next
'loop over all Shell-Windows
For Each GetOpenIEByURL In objShellWindows
    'if the document is of type HTMLDocument, it is an IE window
    If TypeName(GetOpenIEByURL.Document) = "HTMLDocument" Then
        'check the URL
        If GetOpenIEByURL.Document.URL = i_URL Then
            'leave, we found the right window
            Exit Function
        End If
    End If
Next
End Function

```

```

'finds an open IE site by checking the title
Function GetOpenIEByTitle(i_Title As String, _
    Optional ByVal i_ExactMatch As Boolean = True) _
    As SHDocVw.InternetExplorer

```

```

Dim objShellWindows As New SHDocVw.ShellWindows

```

```

If i_ExactMatch = False Then i_Title = "*" & i_Title & "*"
'ignore errors when accessing the document property
On Error Resume Next

```

```

'loop over all Shell-Windows
For Each GetOpenIEByTitle In objShellWindows
    'if the document is of type HTMLDocument, it is an IE window
    If TypeName(GetOpenIEByTitle.Document) = "HTMLDocument" Then
        'check the title
        If GetOpenIEByTitle.Document.Title Like i_Title Then
            'leave, we found the right window
            Exit Function
        End If
    End If
Next
End Function

```

```

Function IE_Preparation(myPageTitle As String, myPageURL As String) _
    As SHDocVw.InternetExplorer

```

```

    'IE object instantiation
    Dim myIE As SHDocVw.InternetExplorer

```

```

    'check if page is already open
    Set myIE = GetOpenIEByTitle(myPageTitle, False)

```

```

    'check if page is already open
    '(does not work if the page is relocated)
    'Set myIE = GetOpenIEByURL(myPageURL)

```

```

    'if the page is not open then try to open it
    If myIE Is Nothing Then
        'page isn't open yet
        'create new IE instance
        Set myIE = GetNewIE
        'make IE window visible
        myIE.Visible = True
        'IE move and resize
        Call IE_MoveAndResize(myIE, 0, 0, 1200, 900)
        Excel.Application.Wait (Now + TimeValue("0:00:01"))
        'load page
        If LoadWebPage(myIE, myPageURL) = False Then
            'page wasn't loaded
            MsgBox "Couldn't open page"
            Exit Function
        End If
        'wait until SmC is loaded
        Excel.Application.Wait (Now + TimeValue("0:00:30"))
    End If

```

```

    'move and resize the window (do it before you start the URL)
    'Dim nhWnd As Long
    'nhWnd1 = FindWindow(vbNullString, "SmartChoice - Windows Internet Explorer provided by
Syngenta")
    'nhWnd1 = myIE.hWnd
    'If nhWnd1 <> 0 Then
    '    MoveWindow nhWnd1, 0, 0, 1200, 900, 1

```

'End If

Set IE\_Preparation = myIE  
End Function

Function IE\_MoveAndResize(ByRef IE As SHDocVw.InternetExplorer, \_  
top As Variant, left As Variant, \_  
width As Variant, height As Variant)  
'move and resize the window  
Dim nhWnd As Long  
nhWnd = IE.hwnd  
If nhWnd <> 0 Then  
MoveWindow nhWnd, top, left, width, height, 1  
End If

End Function

Function z\_CloseIE(strWindowTitle As String)  
Const WM\_CLOSE = &H10  
Dim hwnd As Long  
hwnd = z\_FindWindowHandle(strWindowTitle)  
If hwnd <> 0 Then  
'bring to the foreground  
SetForegroundWindow hwnd  
'close the IE window  
PostMessage hwnd, WM\_CLOSE, 0&, 0&  
End If  
End Function

Function z\_CloseIE2(hwnd As Long)  
Const WM\_CLOSE = &H10  
If hwnd <> 0 Then  
'bring to the foreground  
SetForegroundWindow hwnd  
'close the IE window  
PostMessage hwnd, WM\_CLOSE, 0&, 0&  
End If  
End Function

Function FindPopUpWindow(optionalParentName As String, Optional hwndParent As Long) As Long  
Dim WindowName As String  
WindowName = vbNullString  
If hwndParent = 0 Then  
hwndParent = z\_FindWindowHandle(optionalParentName)  
End If  
Dim hWndChild As Long  
hWndChild = FindWindowEx(hwndParent, 0&, vbNullString, WindowName)  
FindPopUpWindow = hWndChild  
'when found do the clicks  
'RetVal = SetForegroundWindow(hwnd)  
'Excel.Application.Wait (Now + TimeValue("0:00:02")) ' this value may need to be adjusted  
'SendMessage hchildwnd, BM\_CLICK, 0, 0

End Function

```
Function z_MoveWbSheetsIntoAnotherWb(Wb_ref As Workbook)
    Dim Wb As Workbook
    For Each Wb In Wb_ref.Application.Workbooks ' Workbooks
        'move all Windows to Wb_ref
        If Wb.name <> Wb_ref.name Then
            If Wb.name <> "PERSONAL.XLSB" Then
                If Wb.name <> "API_VBA_GenerateSmartchoiceReports.xlsb" Then
                    Windows(Wb.name).Activate
                    Sheets(left(Wb.name, 31)).Select
                    Sheets(left(Wb.name, 31)).Move After:=Wb_ref.Sheets(Sheets.Count)
                End If
            End If
        End If
    Next
    'For Each Wb In Wb_ref.Application.Workbooks
    '    If Wb.Name = "PERSONAL.XLSB" Then
    '        Windows(Wb.Name).Visible = False
    '    End If
    'Next
End Function
```

```
Function Key_Alt_O()
    'http://www.unet.univie.ac.at/~a7425519/programme/hex2dez.htm
    Const VK_MENU = 18 '0x12
    Const VK_O = 79 '0x4F
    'WinKey down
    keybd_event VK_MENU, 0, 0, 0
    'F key down
    keybd_event VK_O, 0, 0, 0
    'M key up
    keybd_event VK_O, 0, KEYEVENTF_KEYUP, 0
    'WinKey up
    keybd_event VK_MENU, 0, KEYEVENTF_KEYUP, 0
End Function
```

```
Function z_ExcelDownloadExists(Wb_ref As Workbook) As Boolean
    Dim Wb As Workbook
    z_ExcelDownloadExists = False
    For Each Wb In Wb_ref.Application.Workbooks
        If Wb.name <> Wb_ref.name Then
            If Wb.name <> "PERSONAL.XLSB" Then
                If Not Wb.name Like "OwnSmCReport*" Then
                    If Not Wb.name Like "SmC*" Then
                        Wb.Application.Windows(Wb.name).Activate
                        z_ExcelDownloadExists = True
                        Exit Function
                    End If
                End If
            End If
        End If
    End If
End Function
```

```

        End If
    End If
Next
Exit Function
End Function

```

Function z\_MoveWbSheetsIntoAnotherWb\_V2(Wb\_ref As Workbook, name As String, Namelter As Integer)

```

    Dim Wb As Workbook
    'All new workbooks are moved
    'If in the last iteration the new wb was created too late it is moved in the next iteration
    'So it may be possible to have included more than one with the same Namelter
    'In that case an error is thrown and a random number is added to the name
    For Each Wb In Wb_ref.Application.Workbooks
        If Wb.name <> Wb_ref.name Then
            If Wb.name <> "PERSONAL.XLSB" Then
                If Wb.name <> "API_VBA_GenerateSmartchoiceReports.xlsb" Then
                    If Wb.name <> "OwnSmCReport_Part2.xlsb" Then
                        If Wb.name Like "SmC*" Then

                            Else
                                Wb.Application.Windows(Wb.name).Activate
                                Wb.Sheets(1).Select
                                'Sheets(left(Wb.Name, 31)).Select
                                Wb.Sheets(1).Move After:=Wb_ref.Sheets(Sheets.Count)
                                'Sheets(left(Wb.Name, 31)).Move After:=Wb_ref.Sheets(Sheets.Count)
                                On Error GoTo OtherName
                                If Namelter < 10 Then
                                    ActiveSheet.name = name & "0" & Namelter
                                Else
                                    ActiveSheet.name = name & Namelter
                                End If
                                On Error GoTo 0
                            End If
                        End If
                    End If
                End If
            End If
        End If
    Next
Exit Function
OtherName:
    Dim errNo As Integer
    errNo = Int((10000 * Rnd) + 1) 'random number from 1 to 10'000
    'wrong if the download is from the project report. Correct manually to Smc_P_VP_
    If name = "SmC_A_VP_" Then
        ActiveSheet.name = "SmC_A_VP_" & Namelter & CStr(errNo)
    ElseIf name = "SmC_P_VP_" Then
        ActiveSheet.name = "SmC_P_VP_" & Namelter & CStr(errNo)
    End If
End Function

```

Function z\_CloseIE3(strWindowTitle As String, Sh\_log As String, VirtualPortfolio\_DropdownPos\_Iter As Integer, \_

```

        FailedDownloads_iter As Integer)
'try to find the IE
Dim My_IE_XL_HTTP500_hWnd As Long
My_IE_XL_HTTP500_hWnd = z_FindWindowHandle(strWindowTitle)
If My_IE_XL_HTTP500_hWnd <> 0 Then
    'IE found, close the IE
    Call z_CloseIE2(My_IE_XL_HTTP500_hWnd)
    DoEvents
    Excel.Application.Wait (Now + TimeValue("0:00:01"))
    'write into the logfile
    'On Error Resume Next
    'mywb.Worksheets(Sh_log).Cells(FailedDownloads_iter + 1, 1) =
VirtualPortfolio_DropdownPos_iter + 1
    'mywb.Worksheets(Sh_log).Cells(FailedDownloads_iter + 1, 2) = "HTTP500"
    'mywb.Worksheets(Sh_log).Cells(FailedDownloads_iter + 1, 3) = Now()
    'On Error GoTo 0
End If
End Function

Private Sub MoveWbSheetsIntoAnotherWb()
    Dim mywb As Workbook
    Dim WbPath As String
    Dim WbName As String
    WbPath = "C:\Users\t740698\Desktop\"
    'wbname = "SmC_Download" & "_" & VBA.DateTime.Day(Now()) & "_" & _
    '    VBA.DateTime.Month(Now()) & "_" & VBA.DateTime.Year(Now()) & ".xlsb"
    WbName = "SmC_Download_07112011"
    'open or activate mywb
    Call z_WorkbookNewOrOpenOrActivate(WbName, WbPath, mywb)
    'bring the Excel session into xlnormal
    Call z_ExcelSessionWindowNormal(mywb)
    Call z_ExcelWorkbookWindowMaximized(mywb)
    'rename logfile
    mywb.Sheets("Sheet1").Select
    mywb.Sheets("Sheet1").name = "Logfile"
    'delete empty sheets
    Call z_DeleteWbSheet(mywb, "Sheet2")
    Call z_DeleteWbSheet(mywb, "Sheet3")
    'move all windows into mywb
    Call z_MoveWbSheetsIntoAnotherWb(mywb)
    'save workbook
    Call z_ExcelWorkbookWindowMaximized(mywb)
    mywb.Save
End Sub

Function z_Wait2(milliseconds As Long)
    Call sleep(milliseconds)
End Function

Function z_Wait(xlapp As Excel.Application, Time_)
    Dim a As Long
    Dim i As Long
    On Error GoTo ErrHandler:

```

```

xlapp.Wait (Now + TimeValue(Time_))
Exit Function
ErrorHandler:
a = 10000000
For i = 1 To a
a = VBA.Sqr(a)
Next
Resume
End Function

```

## **File: Sudoku\_VersionNeutral**

### **Private Sub Workbook\_Open()**

```

'DieseArbeitsmappe>
'Algorithmus: Beim öffnen der Excel Arbeitsmappe öffnen sich zwei Dialogboxen, welche je einen
'Eingabewert verlangen in den Variablen myInput und myInput1 speichern und entsprechend dem
Wert
'reagiert das Programm anders.
Dim myInput1
Tabelle3.Activate
MsgBox ("Dieses Sudoku-Analysetool mit Solver ist ein Geschenk von Roli ")
myInput1 = Application.InputBox("Bitte geben Sie ein Y ein, um zu bestätigen, dass Sie das
Programm nicht vervielfältigen und nur zum privaten Gebrauch benützen.", , "N", , , , 2)
If myInput1 = "Y" Then

Else
ActiveWorkbook.Close SaveChanges:=False
End If

Dim myInput
Tabelle5.Activate
myInput = Application.InputBox("Willkommen beim Sudoku Analysetool. Möchten Sie eine
Einführung in das Tool?", , "Y", 430, 140, , , 2)

If myInput = "Y" Then
Tabelle6.Activate
Worksheets("Hilfetext").Cells(1, 1).Activate
Else
Tabelle1.Activate
Worksheets("Spielfelder").Cells(1, 1).Activate
End If
End Sub

```

### **Private Sub CommandButton5\_Click()**

```

'Tabelle1(Spielfelder)>
'Algorithmus: Beim klicken des CommandButton5 mit dem Namen Start öffnet sich das UserForm1

```

```
UserForm1.Show
```

```
End Sub
```

```
Private Sub CommandButton1_Click()
```

```
'UserForm1>Vorbereitung>Alle Felder auf Null setzen
```

```
  Sheets("Spielfelder").Select
```

```
  ActiveWindow.ScrollRow = 1
```

```
  Dim Temp
```

```
  Temp = Application.InputBox(prompt:="Beide oder nur Lösung auf Null setzen? Y: für Beide; N: für  
Lösung", Default:="N", Type:=2)
```

```
  If Temp = "Y" Then
```

```
    Worksheets("Spielfelder").Range("B11:J19,L11:T19").ClearContents
```

```
    Worksheets("Spielfelder").Range("B11:J19,L11:T19").Font.Size = 10
```

```
  Else
```

```
    Worksheets("Spielfelder").Range("B11:J19").ClearContents
```

```
    Worksheets("Spielfelder").Range("B11:J19").Font.Size = 10
```

```
  End If
```

```
  Call NullSetzen
```

```
End Sub
```

## **Function NullSetzen()**

```
'UserForm1>Vorbereitung>Alle Felder auf Null setzen>CommandButton1_Click()
```

```
  Worksheets("Tabellen").Range("C10:K99").Value = 0
```

```
  Worksheets("Tabellen").Range("N11:V19").Value = 0
```

```
  Worksheets("Tabellen").Range("N21:V29").Value = 0
```

```
  Worksheets("Tabellen").Range("N31:V39").Value = 0
```

```
  Worksheets("Tabellen").Range("N41:V49").Value = 0
```

```
  Worksheets("Tabellen").Range("N55:V63").Value = 0
```

```
  Worksheets("Tabellen").Range("N68:V70").Value = 0
```

```
  Worksheets("Tabellen").Range("N74:V76").Value = 0
```

```
  Worksheets("Tabellen").Range("N81:V83").Value = 0
```

```
  Worksheets("Tabellen").Range("N88:V91").Value = 0
```

```
End Function
```

## **Private Sub CommandButton8\_Click()**

```
'UserForm1>Vorbereitung>gegebene Werte eingeben
```

```
  Sheets("Spielfelder").Select
```

```
  ActiveWindow.ScrollRow = 1
```

```
  Dim Temp
```

```
  Temp = Application.InputBox(prompt:="Stimmen die Werte in der Tabelle rechts? Wenn Nein  
geben Sie N ein, passen die Werte in der Tabelle an und drücken die Start Taste von neuem.",  
Default:="Y", Type:=2)
```

```
  If Temp = "Y" Then
```

```
    Worksheets("Spielfelder").Range("B11:J19") =
```

```
Worksheets("Spielfelder").Range("L11:T19").Value
```

```
    Worksheets("Spielfelder").Range("B11:J19").Font.Color = vbBlack
```



```
Else
    UserForm1.Hide
End If
```

**End Sub**

### **Private Sub CommandButton2\_Click()**

```
'UserForm1>Step by Step>Kandidaten bestimmen
    Sheets("Tabellen").Select
    ActiveWindow.ScrollRow = 1
    UserForm1.Left = 5
    UserForm1.Top = 1
```

```
    Call KandidatenBestimmen
```

**End Sub**

### **Function KandidatenBestimmen()**

```
'UserForm1>Step by Step>Kandidaten bestimmen>CommandButton2_Click()
'Algorithmus: Scannt das Tabellenblatt "Spielfeld" und prüft dabei für jedes Feld (y,x) welche Zahlen
'nicht mehr möglich sind. Im Datenblatt "Tabellen" werden diese nicht mehr möglichen Zahlen für
jedes
'(y,x) als z1=1, z2=2 ... z9=9 gesetzt. Solange ein zi den Startwert Null besitzt ist dieser Wert zi
'für (y,x) noch möglich.
    Call SetzeZWerteFuerGegebeneFelder
    Call SetzeZWerteFuerSpalten
    Call SetzeZWerteFuerZeilen
    Call SetzeZWerteFuerBloেকে
```

**End Function**

### **Function SetzeZWerteFuerGegebeneFelder()**

```
'UserForm1>Step by Step>Kandidaten
bestimmen>CommandButton2_Click()>KandidatenBestimmen()
'Algorithmus: wenn im Tabellenblatt "Spielfeld" (y,x)=(int2,int1) ungleich Leer, dann setze
'im Tabellenblatt "Tabellen" für(y=int2,x=int1)(z1=1,z2=2...z9=9)
    Dim itn1, itn2 As Integer
    For itn1 = 2 To 10
        For itn2 = 11 To 19
            Wert = Worksheets("Spielfelder").Cells(itn2, itn1).Value
            If Wert > 0 Then
                Worksheets("Tabellen").Cells(10 + 1 + (itn2 - 11) * 10, 1 + itn1).Value = 1
                Worksheets("Tabellen").Cells(10 + 2 + (itn2 - 11) * 10, 1 + itn1).Value = 2
                Worksheets("Tabellen").Cells(10 + 3 + (itn2 - 11) * 10, 1 + itn1).Value = 3
                Worksheets("Tabellen").Cells(10 + 4 + (itn2 - 11) * 10, 1 + itn1).Value = 4
                Worksheets("Tabellen").Cells(10 + 5 + (itn2 - 11) * 10, 1 + itn1).Value = 5
                Worksheets("Tabellen").Cells(10 + 6 + (itn2 - 11) * 10, 1 + itn1).Value = 6
                Worksheets("Tabellen").Cells(10 + 7 + (itn2 - 11) * 10, 1 + itn1).Value = 7
                Worksheets("Tabellen").Cells(10 + 8 + (itn2 - 11) * 10, 1 + itn1).Value = 8
                Worksheets("Tabellen").Cells(10 + 9 + (itn2 - 11) * 10, 1 + itn1).Value = 9
            End If
        Next itn2
    Next itn1
```

Next itn1

## End Function

### Function SetzeZWerteFuerSpalten()

'UserForm1>Step by Step>Kandidaten

bestimmen>CommandButton2\_Click()>KandidatenBestimmen()

'Algorithmus: Setze im Tabellenblatt "Spielfeld"  $(y,x)=(Y,X)$  und scanne die Zellen der

'Spalten  $(y,x)=(i,X)$  nach Zahlen K ungleich Null. Setze im Tabellenblatt "Tabellen" im Feld

' $(y=a1+10*Cnt3+K,x=b1+Cnt2)$  den Wert  $zk=K$

Dim X, Y, i, K, a1, b1, Cnt1, Cnt2, Cnt3 As Integer

Cnt3 = 0

a1 = 10

b1 = 3

For Y = 11 To 19

    Cnt2 = 0

    For X = 2 To 10

        Cnt1 = 0

        For i = 11 To 19

            K = Worksheets("Spielfelder").Cells(i, X).Value

            If K > 0 Then

                Worksheets("Tabellen").Cells(a1 + K + Cnt3 \* 10, b1 + Cnt2).Value = K

            End If

            Cnt1 = Cnt1 + 1

        Next i

        Cnt2 = Cnt2 + 1

    Next X

    Cnt3 = Cnt3 + 1

Next Y

## End Function

### Function SetzeZWerteFuerZeilen()

'UserForm1>Step by Step>Kandidaten

bestimmen>CommandButton2\_Click()>KandidatenBestimmen()

'Algorithmus: Setze im Tabellenblatt "Spielfeld"  $(y,x)=(Y,X)$  und scanne die Zellen der

'Zeilen  $(y,x)=(Y,i)$  nach Zahlen K ungleich Null. Setze im Tabellenblatt "Tabellen" im Feld

' $(y=a1+10*Cnt3+K,x=b1+Cnt2)$  den Wert  $zk=K$

Dim X, Y, i, K, a1, b1, Cnt1, Cnt2, Cnt3 As Integer

Cnt3 = 0

a1 = 10

b1 = 3

For Y = 11 To 19

    Cnt2 = 0

    For X = 2 To 10

        Cnt1 = 0

        For i = 2 To 10

            K = Worksheets("Spielfelder").Cells(Y, i).Value

            If K > 0 Then

                Worksheets("Tabellen").Cells(a1 + K + Cnt3 \* 10, b1 + Cnt2).Value = K

```

End If
Cnt1 = Cnt1 + 1
Next i
Cnt2 = Cnt2 + 1
Next X
Cnt3 = Cnt3 + 1
Next Y

```

## End Function

### Function SetzeZWerteFuerBloেকে()

'UserForm1>Step by Step>Kandidaten  
bestimmen>CommandButton2\_Click()>KandidatenBestimmen()  
'Algorithmus: Setze im Tabellenblatt "Spielfelder" (a3,b3) abwechselungsweise gleich dem Feld  
'links oben der verschiedenen neun Blöcke. Scanne die Zellen der Blöcke mit Startwerten(a3,b3)  
'und speichere den Wert der Zelle in der Matrix Blk(a2,b2) wobei a2 die Blocknummer 1 bis 9  
'beschreibt und b2 die Zellennummer 1 bis 9 innerhalb des Blocks.  
'Setze im Tabellenblatt "Spielfelder" (a3,b3) abwechselungsweise gleich dem Feld links oben der  
'verschiedenen neun Blöcke um die Matrix Bkl(count,i) mit den Laufvariablen count und i auszulesen  
'und den Wert in K zu speichern. Für Werte von K ungleich Null Setze im Tabellenblatt "Tabellen"  
'im Feld  $(y=a3-10+cnt4-1)*10+K$ ,  $x=b3+cnt5$  den Wert  $zk=K$ .

```

Dim Blk(10, 10) As Integer
Dim a2, b2, a3, b3, cnt4, cnt5
a2 = 1

```

```

For Count = 1 To 9
  If Count = 1 Then
    a3 = 11
    b3 = 2
  ElseIf Count = 2 Then
    a3 = 11
    b3 = 5
  ElseIf Count = 3 Then
    a3 = 11
    b3 = 8
  ElseIf Count = 4 Then
    a3 = 14
    b3 = 2
  ElseIf Count = 5 Then
    a3 = 14
    b3 = 5
  ElseIf Count = 6 Then
    a3 = 14
    b3 = 8
  ElseIf Count = 7 Then
    a3 = 17
    b3 = 2
  ElseIf Count = 8 Then
    a3 = 17
    b3 = 5
  Else

```

```
a3 = 17  
b3 = 8
```

```
End If
```

```
cnt4 = 1
```

```
b2 = 1
```

```
While cnt4 < 4
```

```
    cnt5 = 1
```

```
    While cnt5 < 4
```

```
        Blk(a2, b2) = Worksheets("Spielfelder").Cells(a3, b3 + cnt5 - 1).Value
```

```
        cnt5 = cnt5 + 1
```

```
        b2 = b2 + 1
```

```
    Wend
```

```
    cnt4 = cnt4 + 1
```

```
    a3 = a3 + 1
```

```
Wend
```

```
a2 = a2 + 1
```

```
Next Count
```

```
For Count = 1 To 9
```

```
    If Count = 1 Then
```

```
        a3 = 11
```

```
        b3 = 2
```

```
    ElseIf Count = 2 Then
```

```
        a3 = 11
```

```
        b3 = 5
```

```
    ElseIf Count = 3 Then
```

```
        a3 = 11
```

```
        b3 = 8
```

```
    ElseIf Count = 4 Then
```

```
        a3 = 14
```

```
        b3 = 2
```

```
    ElseIf Count = 5 Then
```

```
        a3 = 14
```

```
        b3 = 5
```

```
    ElseIf Count = 6 Then
```

```
        a3 = 14
```

```
        b3 = 8
```

```
    ElseIf Count = 7 Then
```

```
        a3 = 17
```

```
        b3 = 2
```

```
    ElseIf Count = 8 Then
```

```
        a3 = 17
```

```
        b3 = 5
```

```
    Else
```

```
        a3 = 17
```

```
        b3 = 8
```

```
    End If
```

```
cnt4 = 1
```

```
While cnt4 < 4
```

```

cnt5 = 1
While cnt5 < 4

    For i = 1 To 9
        K = Blk(Count, i)
        If K > 0 Then
            Worksheets("Tabellen").Cells((a3 - 10 + cnt4 - 1) * 10 + K, b3 + cnt5).Value = K
        End If
        Cnt1 = Cnt1 + 1
    Next i

    cnt5 = cnt5 + 1
Wend
cnt4 = cnt4 + 1
Wend
Next Count

```

**End Function**

### **Private Sub CommandButton3\_Click()**

```

'UserForm1>Step by Step>Kandidaten analysieren
'Algorithmus: Selektiere das Tabellenblatt "Tabellen" und rufe die Funktion
'KandidatenAnalysieren auf
    Sheets("Tabellen").Select
    UserForm1.Left = 5
    UserForm1.Top = 1

```

```

    ActiveWindow.ScrollRow = 10
    Call KandidatenAnalysieren

```

**End Sub**

### **Function KandidatenAnalysieren()**

```

'UserForm1>Step by Step>Kandidaten analysieren>CommandButton3_Click()
'Algorithmus: Auswertung der linken Seite des Tabellenblattes "Tabellen" und rausschreiben der
'Resultate auf die rechte Seite in die Matrizen. Die Matrizen beinhalten die Anzahl Z in den
'Dimensionen (zi,xi) für den Spaltentest, (zi,yi) für den Zeilentest, (zi,Blki) für den Blocktest
'und (yi,xi) für den Feldtest
    Call Spaltentests
    Call Zeilentests
    Call Blocktests
    Call Feldtests

```

**End Function**

### **Function Spaltentests()**

```

'UserForm1>Step by Step>Kandidaten
analysieren>CommandButton3_Click()>KandidatenAnalysieren()
'Algorithmus: Setze im Tabellenblatt "Tabelle" linke Seite (z,x) d.h. eine Spalte X und einen
'Wert Z. Scanne für jedes (z,x) durch die neun Zeilen Y, speichere den Z-Wert in der Variablen
'Wert und summiere die neun Z-Werte in der Variablen Sum. Dividieren der Summe durch den z-Wert

```

*'ergibt die Anzahl zi in der Spalte x. Speichern des Wertes der Variablen Anz in der Matrix  
'Spaltentest(z,x) und rausschreiben auf die rechte Seite des Tabellenblattes "Tabellen" in die  
'entsprechende Matriz.*

```
Dim Wert, Sum, Anz, Spaltentest(9, 9) As Integer  
Wert = 0
```

```
For X = 3 To 11  
  For z = 11 To 19  
    Anz = 0  
    Sum = 0  
    For Y = 1 To 9  
      Wert = Worksheets("Tabellen").Cells(z + 10 * (Y - 1), X)  
      Sum = Sum + Wert  
    Next Y  
    Anz = Sum / (z - 10)  
    Spaltentest(z - 11, X - 3) = Anz  
  Next z  
Next X  
Worksheets("Tabellen").Range("N11:V19") = Spaltentest
```

**End Function**

### Function Zeilentests()

*'UserForm1>Step by Step>Kandidaten*

*analysieren>CommandButton3\_Click()>KandidatenAnalysieren()*

*'Algorithmus: Setze im Tabellenblatt "Tabelle" linke Seite (z,y) d.h. eine Zeile Y und einen*

*'Wert Z. Scanne für jedes (z,y) durch die neun Spalten X, speichere den Z-Wert in der Variablen*

*'Wert und summiere die neun Z-Werte in der Variablen Sum. Dividieren der Summe durch den z-Wert*

*'ergibt die Anzahl zi in der Zeile y. Speichern des Wertes der Variablen Anz in der Matrix*

*'Zeilentest(z,y) und rausschreiben auf die rechte Seite des Tabellenblattes "Tabellen" in die*

*'entsprechende Matriz.*

```
Dim Zeilentest(9, 9) As Integer  
For Y = 1 To 9  
  For z = 11 To 19  
    Anz = 0  
    Sum = 0  
    For X = 3 To 11  
      Wert = Worksheets("Tabellen").Cells(z + 10 * (Y - 1), X)  
      Sum = Sum + Wert  
    Next X  
    Anz = Sum / (z - 10)  
    Zeilentest(z - 11, Y - 1) = Anz  
  Next z  
Next Y  
Worksheets("Tabellen").Range("N21:V29") = Zeilentest
```

**End Function**

### Function Blocktests()

*'UserForm1>Step by Step>Kandidaten*

*analysieren>CommandButton3\_Click()>KandidatenAnalysieren()*

*'Algorithmus: Setze im Tabellenblatt "Tabelle" die linke Seite (z,Bkl) d.h. einen Block Bkl und*

*'einen Wert Z. Scanne für jedes (z,Blk) durch die neun Felder des Blocks, speichere den Z-Wert*

*'in der Variablen Wert und summiere die neun Z-Werte in der Variablen Sum. Dividieren der Summe*

*'durch den z-Wert ergibt die Anzahl zi im Block Blk. Speichern des Wertes der Variablen Anz  
'in der Matrix Blocktest(z,Blk)und rausschreiben auf die rechte Seite des Tabellenblattes  
"Tabellen" in die entsprechende Matriz.*

Dim Blocktest(9, 9) As Integer

For Count = 1 To 9

    If Count = 1 Then

        a3 = 11

        b3 = 2

    Elseif Count = 2 Then

        a3 = 11

        b3 = 5

    Elseif Count = 3 Then

        a3 = 11

        b3 = 8

    Elseif Count = 4 Then

        a3 = 14

        b3 = 2

    Elseif Count = 5 Then

        a3 = 14

        b3 = 5

    Elseif Count = 6 Then

        a3 = 14

        b3 = 8

    Elseif Count = 7 Then

        a3 = 17

        b3 = 2

    Elseif Count = 8 Then

        a3 = 17

        b3 = 5

    Else

        a3 = 17

        b3 = 8

End If

For z = 1 To 9

Anz = 0

Sum = 0

cnt4 = 1

    While cnt4 < 4

        cnt5 = 1

        While cnt5 < 4

            Wert = Worksheets("Tabellen").Cells((a3 - 10 + cnt4 - 1) \* 10 + z, b3 + cnt5)

            Sum = Sum + Wert

            cnt5 = cnt5 + 1

        Wend

        cnt4 = cnt4 + 1

    Wend

Anz = Sum / z

Blocktest(z - 1, Count - 1) = Anz

Next z

Next Count

Worksheets("Tabellen").Range("N31:V39") = Blocktest

## End Function

### Function Feldtests()

```
'UserForm1>Step by Step>Kandidaten  
analysieren>CommandButton3_Click()>KandidatenAnalysieren(  
'Algorithmus: Setze im Tabellenblatt "Tabelle" linke Seite ein Feld (y,x)und scanne für jedes  
'(y,x) durch die neun Werte x, speichere den Z-Wert in der Variablen Wert und summiere die neun  
'Z-Werte in der Variablen Sum. Dividieren der Summe durch den z-Wert ergibt die Anzahl zi in  
'der Spalte x. Speichern des Wertes der Variablen Anz in der Matrix Feldtest(y,x) und rausschreiben  
'auf die rechte Seite des Tabellenblattes "Tabellen" in die entsprechende Matrizze.  
Dim Feldtest(9, 9) As Integer  
For Y = 1 To 9  
    For X = 3 To 11  
        Anz = 0  
        Sum = 0  
        For z = 11 To 19  
            Wert = Worksheets("Tabellen").Cells(z + 10 * (Y - 1), X)  
            If Wert > 0 Then  
                Sum = Sum + 1  
            End If  
        Next z  
        Anz = Sum  
        Feldtest(Y - 1, X - 3) = Anz  
    Next X  
Next Y  
Worksheets("Tabellen").Range("N41:V49") = Feldtest
```

## End Function

### Private Sub CommandButton4\_Click()

```
'UserForm1>Step by Step>Lösungswert und Index bestimmen  
'Algorithmus: Tabellenblatt "Tabellen" selektieren und die Funktion  
'LoesungswertUndIndexBestimmen(1) aufrufen  
Sheets("Tabellen").Select  
Sheets("Tabellen").Select  
ActiveWindow.ScrollRow = 1  
UserForm1.Left = 5  
UserForm1.Top = 1  
  
ActiveWindow.ScrollRow = 52
```

```
Call LoesungswertUndIndexBestimmen(1)
```

## End Sub

### Function LoesungswertUndIndexBestimmen(Farbe)

```
'UserForm1>Step by Step>Lösungswert und Index bestimmen>CommandButton4_Click()  
Call SucheWerteAcht  
Call BestimmeIndexInSpalten(Farbe)  
Call BestimmeIndexInZeilen(Farbe)  
Call BestimmeIndexInBloecken(Farbe)  
Call BestimmeIndexInFeldern(Farbe)
```



'weitere Auswertungslogik hier einbauen

'=====

## **End Function**

### **Function SucheWerteAcht()**

*'UserForm1>Step by Step>Lösungswert und Index bestimmen>CommandButton4\_Click()*

*'>LoesungswertUndIndexBestimmen(Farbe)*

*'Algorithmus:Scannt die Matrizen Zeilen, Spalten, Blöcke auf der rechten Seite im Tabellenblatt*

*"Tabellen" von links nach rechts und Zeile für Zeile durch. Jede Matrize hat 81 Felder und*

*'Cnt1 nimmt Werte von 1 bis 81 an falls an entsprechender Stelle ein Wert c.Value von 8 steht.*

*'Der Wert von Cnt1 wird in eine Matrize rechts unten im Tabellenblatt "Tabellen" rausgeschrieben.*

Dim c As Variant

Dim Cnt1, Cnt2 As Integer

Cnt1 = 1

Cnt2 = 1

For Each c In Worksheets("Tabellen").Range("N11:V19")

    If c.Value = 8 Then

        Worksheets("Tabellen").Cells(68, 13 + Cnt2) = Cnt1

        Cnt2 = Cnt2 + 1

    End If

    Cnt1 = Cnt1 + 1

Next c

Cnt1 = 1

Cnt2 = 1

For Each c In Worksheets("Tabellen").Range("N21:V29")

    If c.Value = 8 Then

        Worksheets("Tabellen").Cells(69, 13 + Cnt2) = Cnt1

        Cnt2 = Cnt2 + 1

    End If

    Cnt1 = Cnt1 + 1

Next c

Cnt1 = 1

Cnt2 = 1

For Each c In Worksheets("Tabellen").Range("N31:V39")

    If c.Value = 8 Then

        Worksheets("Tabellen").Cells(70, 13 + Cnt2) = Cnt1

        Cnt2 = Cnt2 + 1

    End If

    Cnt1 = Cnt1 + 1

Next c

## **End Function**

## Function BestimmeIndexInSpalten(Farbe)

*'UserForm1>Step by Step>Lösungswert und Index bestimmen>CommandButton4\_Click()*

*'>LoesungswertUndIndexBestimmen(Farbe)*

*'Algorithmus:Liest die Indexe der Testmatrizen im Tabellenblatt "Tabellen" ein und bestimmt den*

*'ersten (y,x) Index mit Mod9 welche den Wert Cnt1 durch 9 teilt, den Rest (das gewünschte Resultat*

*'ausser für ein Cnt der neunten Reihe, das gibt Idx=0 und wird durch die If Anweisung erkannt und*

*'korrigiert)in Idx speichert und in eine Matrize weiter unten im Tabellenblatt speichert.*

*'Die Backslash Funktion bestimmt den Ganzzahligen Teiler und vernachlässigt den Restbetrag. Damit*

*'wird der z Wert bestimmt da sich die zi Werte in der Zeile i der Matrix befinden. Dieser Wert wird*

*'weiter unten ins Tabellenblatt geschrieben. Mit der Laufvariablen i wird in der linken Seite des*

*'Tabelleblattes der einzig verbleibende Ort (x,y) gesucht, deren z Wert nicht den Wert wrt besitzt*

*'und somit die gesuchten Koordinaten (y,x) besitzt. Der bestimmte Wert für y wird weiter unten ins*

*'Tabelleblatt geschrieben. Der Wert wird im Tabelleblatt "Spielfelder" an die richtige Koordinate*

*'(y,x) geschrieben.*

Dim Idx, Tmp, Wrt As Integer

Cnt2 = 1

Cnt1 = Worksheets("Tabellen").Cells(68, 13 + Cnt2)

While Cnt1 > 0

    Idx = Cnt1 Mod 9

    Worksheets("Tabellen").Cells(74, 13 + Cnt2) = Idx

    Wrt = (Cnt1 \ 9) + 1

    If Idx = 0 Then

        Idx = 9

        Worksheets("Tabellen").Cells(74, 13 + Cnt2) = Idx

        Wrt = Wrt - 1

    End If

    Worksheets("Tabellen").Cells(81, 13 + Cnt2) = Wrt

For i = 1 To 9

    Tmp = Worksheets("Tabellen").Cells(10 \* i + Wrt, 2 + Idx)

    If Tmp = Wrt Then

    Else

        Worksheets("Tabellen").Cells(88, 13 + Cnt2) = i

        Worksheets("Spielfelder").Cells(10 + i, 1 + Idx) = Wrt

        If Farbe = 1 Then

            Worksheets("Spielfelder").Cells(10 + i, 1 + Idx).Font.Color = RGB(0, 0, 255)

        Elseif Farbe = 2 Then

            Worksheets("Spielfelder").Cells(10 + i, 1 + Idx).Font.Color = RGB(255, 0, 0)

        Else

            Worksheets("Spielfelder").Cells(10 + i, 1 + Idx).Font.Color = RGB(255, 0, 255)

        End If

    End If

Next i

```

    Cnt2 = Cnt2 + 1
    Cnt1 = Worksheets("Tabellen").Cells(68, 13 + Cnt2)
Wend

```

## End Function

### Function BestimmeIndexInZeilen(Farbe)

```

'UserForm1>Step by Step>Lösungswert und Index bestimmen>CommandButton4_Click()
'>LoesungswertUndIndexBestimmen(Farbe)
'Algorithmus:Liest die Indexe der Testmatrizen im Tabellenblatt "Tabellen" ein und bestimmt
den
'ersten (y,x) Index mit Mod9 welche den Wert Cnt1 durch 9 teilt, den Rest (das gewünschte
Resultat
'ausser für ein Cnt der neunten Reihe, das gibt Idx=0 und wird durch die If Anweisung erkannt
und
'korrigiert)in Idx speichert und in eine Matrize weiter unten im Tabellenblatt speichert.
'Die Backslash Funktion bestimmt den ganzzahligen Teiler und vernachlässigt den
Restbetrag. Damit
'wird der z Wert bestimmt da sich die zi Werte in der Zeile i der Matrix befinden. Dieser Wert
wird
'weiter unten ins Tabellenblatt geschrieben. Mit der Laufvariablen i wird in der linken Seite
des
'Tabellenblattes der einzig verbleibende Ort (y,x) gesucht, deren z Wert nicht den Wert wrt
besitzt
'und somit die gesuchten Koordinaten (y,x) besitzt. Der bestimmte Wert für x wird weiter
unten ins
'Tabellenblatt geschrieben. Der Wert wird im Tabellenblatt "Spielfelder" an die richtige
Koordinate
'(y,x) geschrieben.
    Cnt2 = 1
    Cnt1 = Worksheets("Tabellen").Cells(69, 13 + Cnt2)
    While Cnt1 > 0
        Idx = Cnt1 Mod 9
        Worksheets("Tabellen").Cells(75, 13 + Cnt2) = Idx
        Wrt = (Cnt1 \ 9) + 1
        If Idx = 0 Then
            Idx = 9
            Worksheets("Tabellen").Cells(75, 13 + Cnt2) = Idx
            Wrt = Wrt - 1
        End If
        Worksheets("Tabellen").Cells(82, 13 + Cnt2) = Wrt
        For i = 1 To 9
            Tmp = Worksheets("Tabellen").Cells(10 * Idx + Wrt, 2 + i)
            If Tmp = Wrt Then
                Else
                    Worksheets("Tabellen").Cells(89, 13 + Cnt2) = i
                    Worksheets("Spielfelder").Cells(10 + Idx, 1 + i) = Wrt
                    If Farbe = 1 Then
                        Worksheets("Spielfelder").Cells(10 + Idx, 1 + i).Font.Color = RGB(0, 0, 254)
                    ElseIf Farbe = 2 Then
                        Worksheets("Spielfelder").Cells(10 + Idx, 1 + i).Font.Color = RGB(254, 0, 0)
                    Else
                        Worksheets("Spielfelder").Cells(10 + Idx, 1 + i).Font.Color = RGB(254, 0, 254)

```

End If

End If

Next i

Cnt2 = Cnt2 + 1

Cnt1 = Worksheets("Tabellen").Cells(69, 13 + Cnt2)

Wend

## End Function

### Function BestimmeIndexInBloecken(Farbe)

*'UserForm1>Step by Step>Lösungswert und Index bestimmen>CommandButton4\_Click()*

*'>LoesungswertUndIndexBestimmen(Farbe)*

*'Algorithmus:Liest die Indexe der Testmatrizen im Tabellenblatt "Tabellen" ein und bestimmt den*

*'ersten (y,x) Index mit Mod9 welche den Wert Cnt1 durch 9 teilt, den Rest (das gewünschte Resultat*

*'ausser für ein Cnt der neunten Reihe, das gibt Idx=0 und wird durch die If Anweisung erkannt und*

*'korrigiert)in Idx speichert und in eine Matrize weiter unten im Tabellenblatt speichert.*

*'Die Backslash Funktion bestimmt den ganzzahligen Teiler und vernachlässigt den Restbetrag. Damit*

*'wird der z Wert bestimmt da sich die zi Werte in der Zeile i der Matrix befinden. Dieser Wert wird*

*'weiter unten ins Tabellenblatt geschrieben. Dann werden auf der linken Seite des Tabellenblattes alle*

*'(y,x) Koordinaten des bestimmten Blockes mit dem bestimmten Wert z gescannt. Der einzig verbleibende*

*'Ort (y,x), deren z Wert nicht den Wert wrt besitzt ist der Ort, der die gesuchten Koordinaten (y,x)*

*'besitzt. Die bestimmten Werte x und y werden weiter unten ins Tabellenblatt geschrieben.*

*Der Wert*

*'wird im Tabellenblatt "Spielfelder" an die richtige Koordinate (y,x) geschrieben.*

Cnt2 = 1

Cnt1 = Worksheets("Tabellen").Cells(70, 13 + Cnt2)

While Cnt1 > 0

Idx = Cnt1 Mod 9

Worksheets("Tabellen").Cells(76, 13 + Cnt2) = Idx

Wrt = (Cnt1 \ 9) + 1

If Idx = 0 Then

Idx = 9

Worksheets("Tabellen").Cells(76, 13 + Cnt2) = Idx

Wrt = Wrt - 1

End If

Worksheets("Tabellen").Cells(83, 13 + Cnt2) = Wrt

Count = Idx

If Count = 1 Then

a3 = 11

b3 = 2

Elseif Count = 2 Then

a3 = 11

```

    b3 = 5
Elseif Count = 3 Then
    a3 = 11
    b3 = 8
Elseif Count = 4 Then
    a3 = 14
    b3 = 2
Elseif Count = 5 Then
    a3 = 14
    b3 = 5
Elseif Count = 6 Then
    a3 = 14
    b3 = 8
Elseif Count = 7 Then
    a3 = 17
    b3 = 2
Elseif Count = 8 Then
    a3 = 17
    b3 = 5
Else
    a3 = 17
    b3 = 8
End If

cnt4 = 1
While cnt4 < 4
    cnt5 = 1
    While cnt5 < 4
        Tmp = Worksheets("Tabellen").Cells((a3 - 10 + cnt4 - 1) * 10 + Wrt, b3 + cnt5)
        If Tmp = Wrt Then
            Else
                Dim X, Y
                X = b3 - 2 + cnt5
                Y = a3 - 10 + cnt4 - 1
                Worksheets("Tabellen").Cells(90, 13 + Cnt2) = X
                Worksheets("Tabellen").Cells(91, 13 + Cnt2) = Y
                Worksheets("Spielfelder").Cells(10 + Y, 1 + X) = Wrt

                If Farbe = 1 Then
                    Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(0, 0, 253)
                Elseif Farbe = 2 Then
                    Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(253, 0, 0)
                Else
                    Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(253, 0, 253)
                End If
            End If
        End If
        cnt5 = cnt5 + 1
    Wend
    cnt4 = cnt4 + 1
Wend
Cnt2 = Cnt2 + 1
Cnt1 = Worksheets("Tabellen").Cells(70, 13 + Cnt2)

```

Wend

## End Function

### Function BestimmeIndexInFeldern(Farbe)

*'UserForm1>Step by Step>Lösungswert und Index bestimmen>CommandButton4\_Click()*

*'>LoesungswertUndIndexBestimmen(Farbe)*

*'Algorithmus:Liest die Indexe der Testmatrizen im Tabellenblatt "Tabellen" ein und bestimmt den*

*'ersten (y,x) Index mit Mod9 welche den Wert Cnt1 durch 9 teilt, den Rest (das gewünschte Resultat*

*'ausser für ein Cnt der neunten Reihe, das gibt Idx=0 und wird durch die If Anweisung erkannt und*

*'korrigiert)in Idx speichert und in eine Matrize weiter unten im Tabellenblatt speichert. Um den*

*'zweiten Index zu bestimmen wird wie folgt vorgegangen: Da Cnt1 in der Zeile y die Werte w1...w9 wie*

*'folgt annimmt (1-1..9;2-10..18;3-19..27;...;9-73..81) wird Cnt1 and die Funktion*

*'Roundup(Cnt1\*(1.1)/10) was folgendes bewirkt (1-0.11..0.99;2-1.1..1.98;3-2.09..2.97;...;9-8.03..8.91)*

*'Somit ist der y Index Idx2 bestimmt. Mit der Laufvariablen i wird in der linken Seite des Tabellenblattes*

*'am Ort (y,x) in Richtung z gescannt und er einzige Wert für z ungleich Null gesucht. Der bestimmte Wert*

*'für z wird weiter unten ins Tabellenblatt und ins Tabellenblatt "Spielfelder" an die richtige Koordinate*

*'(y,x) geschrieben.*

Cnt1 = 1

Cnt2 = 1

Dim c, c1 As Variant

Dim myRan, myRan1 As Variant

Dim Idx2, Cnt3 As Integer

Cnt3 = 1.1

For Each c In Worksheets("Tabellen").Range("N41:V49")

    If c.Value = 8 Then

        Idx = Cnt1 Mod 9

        Idx2 = Application.WorksheetFunction.RoundUp(Cnt3 / 10, 0)

        If Idx = 0 Then

            Idx = 9

        End If

        Cnt2 = 1

        a1 = 11 + (Idx2 - 1) \* 10

        b1 = 2 + Idx

        d1 = 19 + (Idx2 - 1) \* 10

        For i = 0 To 8

            If Worksheets("Tabellen").Cells(a1 + i, b1).Value = 0 Then

                Worksheets("Tabellen").Cells(54 + Idx2, 13 + Idx) = Cnt2

                Worksheets("Spielfelder").Cells(10 + Idx2, 1 + Idx) = Cnt2

                Worksheets("Spielfelder").Cells(10 + Idx2, 1 + Idx).Font.Color = RGB(0, 0, 252)

```

    If Farbe = 1 Then
        Worksheets("Spielfelder").Cells(10 + Idx2, 1 + Idx).Font.Color = RGB(0, 0, 252)
    ElseIf Farbe = 2 Then
        Worksheets("Spielfelder").Cells(10 + Idx2, 1 + Idx).Font.Color = RGB(252, 0, 0)
    Else
        Worksheets("Spielfelder").Cells(10 + Idx2, 1 + Idx).Font.Color = RGB(252, 0, 252)
    End If

    End If
    Cnt2 = Cnt2 + 1
    Next i
End If
Cnt1 = Cnt1 + 1
Cnt3 = Cnt1 + Cnt1 * 0.1
Next c

```

## End Function

### Private Sub CommandButton9\_Click()

*'UserForm1>Step by Step>Analysefelder sichern und Null setzen*

*'Algorithmus: Öffnet Dialogbox bezüglich Speicherung der Ergebnisse ins History. Click auf den Yes*

*'Button für die Speicherung des Bereiches Worksheets("Tabellen").Range("M10:V99").Value an eine freie*

*'Stelle im Tabellenblatt "History" (In die Zeilen 5 bis 33 und die Spalten 2+Temp\*15 bis 11+Temp\*15).*

*'Um die Idee mit dem freien Bereich umzusetzen müsste Temp bestimmt oder übergeben werden. Im Moment*

*'wird immer dieselbe Stelle überschrieben. Am Ende wird die Funktion NullSetzen() aufgerufen, welche*

*'Alle Felder im Tabellenblatt "Tabelle" auf Null setzt.*

```

Sheets("History").Select
ActiveWindow.ScrollRow = 1

```

```

Static Count As Integer
Count = Count + 1
Dim Txt, Antwort

```

```

Txt = "Das war die Iteration Nr." & Count & _
" Fahren Sie bei 'Kandidaten bestimmen' fort" & _
" Möchten Sie die berechneten Ergebnisse ins History speichern?"
Antwort = MsgBox(Txt, vbYesNo)
If Antwort = vbYes Then
    Worksheets("History").Range(Worksheets("History").Cells(5, 2 + Temp * 15), _
    Worksheets("History").Cells(33, 11 + Temp * 15)) = _
    Worksheets("Tabellen").Range("M10:V99").Value
End If

```

```

Sheets("Spielfelder").Select

```

```
UserForm1.Left = 5  
UserForm1.Top = 1
```

```
ActiveWindow.ScrollRow = 1
```

```
Call NullSetzen
```

## End Sub

### Function NullSetzen()

*'UserForm1>Vorbereitung>Alle Felder auf Null setzen>CommandButton1\_Click()*

```
Worksheets("Tabellen").Range("C10:K99").Value = 0  
Worksheets("Tabellen").Range("N11:V19").Value = 0  
Worksheets("Tabellen").Range("N21:V29").Value = 0  
Worksheets("Tabellen").Range("N31:V39").Value = 0  
Worksheets("Tabellen").Range("N41:V49").Value = 0  
Worksheets("Tabellen").Range("N55:V63").Value = 0  
Worksheets("Tabellen").Range("N68:V70").Value = 0  
Worksheets("Tabellen").Range("N74:V76").Value = 0  
Worksheets("Tabellen").Range("N81:V83").Value = 0  
Worksheets("Tabellen").Range("N88:V91").Value = 0
```

## End Function

### Private Sub CommandButton12\_Click()

*'UserForm1>In one Step>Start*

*'Algorithmus: Fragt über eine InputBox nach der Anzahl Iterationen und führt die Funktionen*

*'KandidatenBestimmen(), KandidatenAnalysieren(), LösungswertUndIndexBestimmen(1)*

*iterationsweise*

*'durch bis die eingegebene Anzahl Iterationen beendet ist oder vorher wenn die Funktion*  
*PruefeFortschritt()*

*'einen Wert ungleich Null zurückgibt. Nach jeder Iteration wird der ein Bereich des*  
*Tabellenblattes*

*'Tabellen in das Tabellenblatt "History" gespeichert und die Funktion NullSetzen()*  
*aufgerufen.*

```
Dim Eingabe As Integer
```

```
Eingabe = Application.InputBox(prompt:="Anzahl Iterationen", Default:=15, Type:=1)
```

```
Dim Iteration As Integer
```

```
Iteration = 0
```

```
While Iteration < Eingabe
```

```
Call KandidatenBestimmen
```

```
Call KandidatenAnalysieren
```

```
Call LoesungswertUndIndexBestimmen(1)
```

```
Stopp = PruefeFortschritt
```



If Stopp = 0 Then

Count = Count + 1

Worksheets("History").Range(Worksheets("History").Cells(5, 2 + Temp \* 15), \_  
Worksheets("History").Cells(33, 11 + Temp \* 15)) = \_  
Worksheets("Tabellen").Range("M10:V99").Value

Call NullSetzen

Iteration = Iteration + 1

Else

Iteration = Eingabe

Call NullSetzen

End If

Wend

Sheets("Spielfelder").Select  
ActiveWindow.ScrollRow = 1

## End Sub

### Function PruefeFortschritt()

*'UserForm1>In one Step>Start>CommandButton12\_Click()*

*'Algorithmus: Prüft in den zwei Bereichen des Tabellenblattes "Tabellen" ob sich in den zwei  
'Bereichen Werte ungleich Null befinden. Solange das der Fall ist findet der Algorithmus  
weitere*

*'Lösungen und die Funktion gibt den Wert Null zurück ansonsten den Wert 1.*

Dim Anz, Stopp As Integer

Anz = 0

Stopp = 0

Dim r1, r2 As Range

Set r1 = Worksheets("Tabellen").Range("N68:V70")

Set r2 = Worksheets("Tabellen").Range("N55:V63")

For i = 1 To 3

For j = 1 To 9

Anz = Anz + r1(i, j)

Next j

Next i

For i = 1 To 9

For j = 1 To 9

Anz = Anz + r2(i, j)

Next j

Next i

If Anz = 0 Then

MsgBox "Die letzte Iteration fand keine neuen Werte mehr. Sie können nun eine Hypothese  
eingeben und verifizieren lassen."

Stopp = 1

PruefeFortschritt = Stopp

```
Else  
    PruefeFortschritt = Stopp  
End If
```

## **End Function**

### **Private Sub Image1\_Click()**

```
'UserForm1>Navigation>ButtonLinkeSeite  
    Sheets("Hilfetext").Select  
    ActiveWindow.ScrollRow = 1
```

## **End Sub**

### **Private Sub Image2\_Click()**

```
'UserForm1>Navigation>ButtonLinkeSeite  
    Sheets("Tabellen").Select  
    ActiveWindow.ScrollRow = 1
```

## **End Sub**

### **Private Sub Image3\_Click()**

```
'UserForm1>Navigation>ButtonLinkeSeite  
    Sheets("Spielfelder").Select  
    ActiveWindow.ScrollRow = 1
```

## **End Sub**

### **Private Sub Image4\_Click()**

```
'UserForm1>Navigation>ButtonMitte  
    ActiveWindow.LargeScroll ToLeft:=1
```

## **End Sub**

### **Private Sub Image5\_Click()**

```
'UserForm1>Navigation>ButtonMitte  
  
    'ActiveWindow.LargeScroll Up:=1  
    ActiveWindow.SmallScroll Up:=4
```

## **End Sub**

### **Private Sub Image6\_Click()**

```
'UserForm1>Navigation>ButtonMitte  
    ActiveWindow.LargeScroll ToRight:=1
```

## **End Sub**

### **Private Sub Image7\_Click()**

*'UserForm1>Navigation>ButtonMitte*

*'ActiveWindow.LargeScroll Down:=1*

*ActiveWindow.SmallScroll Down:=4*

### **End Sub**

### **Private Sub Image8\_Click()**

*'UserForm1>Navigation>ButtonRechts*

*'Algorithmus:Setzt das UserForm1 links oben*

UserForm1.Left = 5

UserForm1.Top = 1

UserForm1.Hide

UserForm1.Show

### **End Sub**

### **Private Sub Image9\_Click()**

*'UserForm1>Navigation>ButtonRechts*

*'Algorithmus:Setzt das UserForm1 rechts oben*

UserForm1.Left = 430

UserForm1.Top = 1

UserForm1.Hide

UserForm1.Show

### **End Sub**

### **Function KandidatenAnzeigen()**

*'UserForm1>Hypothesen>KandidatenAnzeigen>CommandButto15\_Click()*

*'Algorithmus: Zeigt im Tabellenblatt "Spielfeld" für alle leeren Felder (y,x) alle zi an, die für diese Felder noch möglich sind. Scannt dabei für jedes Feld (y,x) durch die Dimension z im Tabellenblatt*

*"Tabellen" und schreibt für jedes z das noch einen Wert von Null hat, den zi in die Variable z1 bis z9.*

For X = 1 To 9

For Y = 1 To 9

z1 = ".."

z2 = ".."

z3 = ".."

z4 = ".."

z5 = ".."

z6 = ".."

z7 = ".."

z8 = ".."

z9 = ".."

For z = 1 To 9

If z = 1 Then

```

If Worksheets("Tabellen").Cells((Y * 10) + z, 2 + X).Value = 0 Then
    z1 = z
    Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Value = _
        z1 & " " & z2 & " " & z3 & " " & _
        z4 & " " & z5 & " " & z6 & " " & _
        z7 & " " & z8 & " " & z9 & " "
    Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 8
    Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(255, 0, 0)
End If
Elseif z = 2 Then
    If Worksheets("Tabellen").Cells((Y * 10) + z, 2 + X).Value = 0 Then
        z2 = z
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Value = _
            z1 & " " & z2 & " " & z3 & " " & _
            z4 & " " & z5 & " " & z6 & " " & _
            z7 & " " & z8 & " " & z9 & " "
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 8
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(255, 0, 0)
    End If
Elseif z = 3 Then
    If Worksheets("Tabellen").Cells((Y * 10) + z, 2 + X).Value = 0 Then
        z3 = z
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Value = _
            z1 & " " & z2 & " " & z3 & " " & _
            z4 & " " & z5 & " " & z6 & " " & _
            z7 & " " & z8 & " " & z9 & " "
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 8
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(255, 0, 0)
    End If
Elseif z = 4 Then
    If Worksheets("Tabellen").Cells((Y * 10) + z, 2 + X).Value = 0 Then
        z4 = z
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Value = _
            z1 & " " & z2 & " " & z3 & " " & _
            z4 & " " & z5 & " " & z6 & " " & _
            z7 & " " & z8 & " " & z9 & " "
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 8
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(255, 0, 0)
    End If
Elseif z = 5 Then
    If Worksheets("Tabellen").Cells((Y * 10) + z, 2 + X).Value = 0 Then
        z5 = z
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Value = _
            z1 & " " & z2 & " " & z3 & " " & _
            z4 & " " & z5 & " " & z6 & " " & _
            z7 & " " & z8 & " " & z9 & " "
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 8
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(255, 0, 0)
    End If
Elseif z = 6 Then
    If Worksheets("Tabellen").Cells((Y * 10) + z, 2 + X).Value = 0 Then
        z6 = z
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Value = _

```

```

        z1 & " " & z2 & " " & z3 & " " & _
        z4 & " " & z5 & " " & z6 & " " & _
        z7 & " " & z8 & " " & z9 & " "
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 8
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(255, 0, 0)
    End If
Elseif z = 7 Then
    If Worksheets("Tabellen").Cells((Y * 10) + z, 2 + X).Value = 0 Then
        z7 = z
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Value = _
        z1 & " " & z2 & " " & z3 & " " & _
        z4 & " " & z5 & " " & z6 & " " & _
        z7 & " " & z8 & " " & z9 & " "
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 8
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(255, 0, 0)
    End If
Elseif z = 8 Then
    If Worksheets("Tabellen").Cells((Y * 10) + z, 2 + X).Value = 0 Then
        z8 = z
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Value = _
        z1 & " " & z2 & " " & z3 & " " & _
        z4 & " " & z5 & " " & z6 & " " & _
        z7 & " " & z8 & " " & z9 & " "
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 8
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(255, 0, 0)
    End If
Elseif z = 9 Then
    If Worksheets("Tabellen").Cells((Y * 10) + z, 2 + X).Value = 0 Then
        z9 = z
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Value = _
        z1 & " " & z2 & " " & z3 & " " & _
        z4 & " " & z5 & " " & z6 & " " & _
        z7 & " " & z8 & " " & z9 & " "
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 8
        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = RGB(255, 0, 0)
    End If
End If
Next z
Next Y
Next X
End Function

```

## Function KandidatenAnzeigenRückgängig()

*'UserForm1>Hypothesen>KandidatenAnzeigen>CommandButto15\_Click()*

*'Löscht den Inhalt aller Felder im Tabellenblatt "Spielfelder" deren Eigenschaft Font.Color eines der Werte der vier RGB Farben besitzt.*

```
For X = 1 To 9
```

```
    For Y = 1 To 9
```

```
        If Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Color = _
```

```
        (RGB(255, 0, 0) Or RGB(254, 0, 0) Or _
```

```
        RGB(253, 0, 0) Or RGB(252, 0, 0)) Then
```

```
            Worksheets("Spielfelder").Cells(10 + Y, 1 + X).Font.Size = 10
```

```

        Worksheets("Spielfelder").Cells(10 + Y, 1 + X).ClearContents
    End If
Next Y
Next X

```

## End Function

### Private Sub CommandButton13\_Click()

*'UserForm1>Hypothesen>Hypothese eingeben*

*'Algorithmus: Ruft UserForm2 auf.*

```

    Sheets("Spielfelder").Select

```

```

    ActiveWindow.ScrollRow = 1

```

```

    UserForm2.Show

```

## End Sub

## Dim MyData As DataObject

### Private Sub CommandButton1\_Click()

*'UserForm1>Hypothesen>Hypothese eingeben>UserForm2>OK*

*'Algorithmus: Das UserForm2 enthält drei TextBoxen mit den Namen TextBox1, TextBox2 und TextBox3.*

*'Um die Eingabewerte rauszulesen werden drei Datenobjekte MyDataX, MyDataY und MyDataW verwendet,*

*'über welche die Eingabewerte in die Variablen X,Y und W geschrieben werden. Danach wird der Wert*

*'W in das Tabellenblatt "Spielfelder" an die Stelle (y,x) geschrieben*

```

    UserForm2.Left = 430

```

```

    UserForm2.Top = 1

```

```

    Set MyDataX = New DataObject

```

```

    Set MyDataY = New DataObject

```

```

    Set MyDataW = New DataObject

```

```

    TextBox1.SelStart = 0

```

```

    TextBox1.SelLength = TextBox1.TextLength

```

```

    TextBox1.Copy

```

```

    MyDataX.GetFromClipboard

```

```

    TextBox2.SelStart = 0

```

```

    TextBox2.SelLength = TextBox1.TextLength

```

```

    TextBox2.Copy

```

```

    MyDataY.GetFromClipboard

```

```

    TextBox3.SelStart = 0

```

```

    TextBox3.SelLength = TextBox1.TextLength

```

```

    TextBox3.Copy

```

```

    MyDataW.GetFromClipboard

```

```
Dim X, Y, W As Integer
X = CInt(MyDataX.GetText)
Y = CInt(MyDataY.GetText)
W = CInt(MyDataW.GetText)
```

```
UserForm2.Hide
```

```
Worksheets("Spielfelder").Range(Cells(10 + Y, 1 + X), _
Cells(10 + Y, 1 + X)) = W
Worksheets("Spielfelder").Range(Cells(10 + Y, 1 + X), _
Cells(10 + Y, 1 + X)).Font.Color = RGB(255, 0, 255)
```

**End Sub**

### **Private Sub CommandButton14\_Click()**

*'UserForm1>Hypothesen>Hypothesen überprüfen*

*'Algorithmus: Fragt über eine InputBox nach der Anzahl Iterationen und führt die Funktionen  
'KandidatenBestimmen(), KandidatenAnalysieren(), LösungswertUndIndexBestimmen(1)  
iterationsweise*

*'durch bis die eingegebene Anzahl Iterationen beendet ist oder vorher, wenn die Funktion  
'Plausibilitätsprüfung() oder PruefeFortschritt()einen Wert ungleich Null zurückgibt. Nach  
jeder*

*'Iteration wird der ein Bereich des Tabellenblattes Tabellen in das Tabellenblatt "History"  
gespeichert*

*'und die Funktion NullSetzen() aufgerufen. Bei der letzten Iteration wird die Funktion  
HypothesePruefen()*

*'aufgerufen, welche prüft ob das Sudoku fertig gelöst ist.*

```
Dim Eingabe, Stopp As Integer
```

```
Eingabe = Application.InputBox(prompt:="Anzahl Iterationen", Default:=10, Type:=1)
```

```
Dim Iteration As Integer
```

```
Iteration = 0
```

```
While Iteration < Eingabe
```

```
    Call KandidatenBestimmen
```

```
    Call KandidatenAnalysieren
```

```
    Call LoesungswertUndIndexBestimmen(2)
```

```
    Stopp = Plausibilitätsprüfung
```

```
    If Stopp = 0 Then
```

```
        Stopp = PruefeFortschritt
```

```
        If Iteration = (Eingabe - 1) Then
```

```
            Call HypothesePruefen
```

```
        End If
```

```
    If Stopp = 0 Then
```

Count = Count + 1

Worksheets("History").Range(Worksheets("History").Cells(5, 2 + Temp \* 15), \_  
Worksheets("History").Cells(33, 11 + Temp \* 15)) = \_  
Worksheets("Tabellen").Range("M10:V99").Value

Call NullSetzen

Iteration = Iteration + 1

ElseIf Stopp = 1 Then

Iteration = Eingabe

Call NullSetzen

End If

ElseIf Stopp = 1 Then

Iteration = Eingabe

Call NullSetzen

End If

Wend

Sheets("Spielfelder").Select

ActiveWindow.ScrollRow = 1

## End Sub

## Function Plausibilitätsprüfung()

*'UserForm1>Hypothesen>Hypothesen überprüfen>CommandButton14\_Click()*

*'Algorithmus: Prüft, ob sich durch die Eingabe der Hypothese irgendwo im Tabellenblatt  
"Spielfeld"*

*'ein Widerspruch ergeben hat, indem in einer Spalte, Zeile oder einem Block zweimal dieselbe  
Zahl*

*'erscheint. Die Funktion gibt den Wert Null zurück, wenn sich kein Widerspruch ereignet hat,  
sonst*

*'der Wert 1*

Dim Stopp As Integer

Stopp = PlausibilitätsprüfungSpalten

If Stopp = 0 Then

Stopp = PlausibilitätsprüfungZeilen

If Stopp = 0 Then

Stopp = PlausibilitätsprüfungBlöcke

End If

End If

Plausibilitätsprüfung = Stopp

## End Function



## Function PlausibilitätsprüfungSpalten()

*'UserForm1>Hypothesen>Hypothesen*

*überprüfen>CommandButton14\_Click()>Plausibilitätsprüfung()*

*'Algorithmus: Liest jeden Wert (y,x) nacheinander in Wert1 ein, liest alle Werte derselben Spalte*

*'nacheinander in Wert2 ein und prüft mit Wert1=Wert2, ob in derselben Spalte zwei gleiche Werte*

*'vorkommen(Ausser dem Wert 0). Ist das der Fall erscheint eine MsgBox mit dem Hinweis auf einen Fehler*

*'Die Funktion gibt Null zurück wenn kein widerspruch aufgetaucht ist, ansonsten Eins.*

Dim Stopp As Integer

Stopp = 0

For i = 1 To 9

For j = 1 To 9

Wert1 = Worksheets("Spielfelder").Cells(10 + i, 1 + j)

For K = 1 To 9

If i = K Then

K = K + 1

Else

Wert2 = Worksheets("Spielfelder").Cells(10 + K, 1 + j)

If (Wert1 = Wert2) And (Not (Wert1 = 0)) Then

MsgBox "Fehlerhafte Hypothese " & "x= " & j & "; y= " & i & "; Wert= " & Wert1

K = 9

j = 9

i = 9

Stopp = 1

End If

End If

Next K

Next j

Next i

PlausibilitätsprüfungSpalten = Stopp

## End Function

## Function PlausibilitätsprüfungZeilen()

*'UserForm1>Hypothesen>Hypothesen*

*überprüfen>CommandButton14\_Click()>Plausibilitätsprüfung()*

*'Algorithmus: Liest jeden Wert (y,x) nacheinander in Wert1 ein, liest alle Werte derselben Zeile*

*'nacheinander in Wert2 ein und prüft mit Wert1=Wert2, ob in derselben Zeile zwei gleiche Werte*

*'vorkommen(Ausser dem Wert 0). Ist das der Fall, erscheint eine MsgBox mit dem Hinweis auf einen Fehler*

*'Die Funktion gibt Null zurück wenn kein Widerspruch aufgetaucht ist, ansonsten Eins.*

Dim Stopp As Integer

Stopp = 0

For i = 1 To 9

For j = 1 To 9

Wert1 = Worksheets("Spielfelder").Cells(10 + i, 1 + j)

For K = 1 To 9

If j = K Then

K = K + 1

```

Else
    Wert3 = Worksheets("Spielfelder").Cells(10 + i, 1 + K)
    If (Wert1 = Wert3) And (Not (Wert1 = 0)) Then
        MsgBox "Fehlerhafte Hypothese " & "x= " & j & "; y= " & i & "; Wert= " & Wert1
        K = 9
        j = 9
        i = 9
        Stopp = 1
    End If
End If
Next K
Next j
Next i
PlausibilitätsprüfungZeilen = Stopp
End Function

```

## Function PlausibilitätsprüfungBloেকে()

*'UserForm1>Hypothesen>Hypothesen*

*überprüfen>CommandButton14\_Click()>Plausibilitätsprüfung()*

*'Algorithmus: Liest jeden Wert (y,x) nacheinander in Wert1 ein, liest alle Werte desselben Blocks*

*'nacheinander in Wert2 ein und prüft mit Wert1=Wert2, ob in demselben Block zwei gleiche Werte*

*'vorkommen(Ausser dem Wert 0). Ist das der Fall erscheint eine MsgBox mit dem Hinweis auf einen Fehler*

*'Die Funktion gibt Null zurück wenn kein widerspruch aufgetaucht ist, ansonsten Eins.*

Dim Stopp As Integer

Stopp = 0

For i = 1 To 9

For j = 1 To 9

Wert1 = Worksheets("Spielfelder").Cells(10 + i, 1 + j)

For K = 1 To 9

For l = 1 To 9

If (i = 1 Or i = 2 Or i = 3) And (j = 1 Or j = 2 Or j = 3) And \_  
(K = 1 Or K = 2 Or K = 3) And (l = 1 Or l = 2 Or l = 3) Then

Wert4 = Worksheets("Spielfelder").Cells(10 + K, 1 + l)

Elseif (i = 1 Or i = 2 Or i = 3) And (j = 4 Or j = 5 Or j = 6) And \_  
(K = 1 Or K = 2 Or K = 3) And (l = 4 Or l = 5 Or l = 6) Then

Wert4 = Worksheets("Spielfelder").Cells(10 + K, 1 + l)

Elseif (i = 1 Or i = 2 Or i = 3) And (j = 7 Or j = 8 Or j = 9) And \_  
(K = 1 Or K = 2 Or K = 3) And (l = 7 Or l = 8 Or l = 9) Then

Wert4 = Worksheets("Spielfelder").Cells(10 + K, 1 + l)

Elseif (i = 4 Or i = 5 Or i = 6) And (j = 1 Or j = 2 Or j = 3) And \_  
(K = 4 Or K = 5 Or K = 6) And (l = 1 Or l = 2 Or l = 3) Then

Wert4 = Worksheets("Spielfelder").Cells(10 + K, 1 + l)

Elseif (i = 4 Or i = 5 Or i = 6) And (j = 4 Or j = 5 Or j = 6) And \_  
(K = 4 Or K = 5 Or K = 6) And (l = 4 Or l = 5 Or l = 6) Then

Wert4 = Worksheets("Spielfelder").Cells(10 + K, 1 + l)

Elseif (i = 4 Or i = 5 Or i = 6) And (j = 7 Or j = 8 Or j = 9) And \_  
(K = 4 Or K = 5 Or K = 6) And (l = 7 Or l = 8 Or l = 9) Then

Wert4 = Worksheets("Spielfelder").Cells(10 + K, 1 + l)

```

Elseif (i = 7 Or i = 8 Or i = 9) And (j = 1 Or j = 2 Or j = 3) And _
    (K = 7 Or K = 8 Or K = 9) And (l = 1 Or l = 2 Or l = 3) Then
    Wert4 = Worksheets("Spielfelder").Cells(10 + K, 1 + l)
Elseif (i = 7 Or i = 8 Or i = 9) And (j = 4 Or j = 5 Or j = 6) And _
    (K = 7 Or K = 8 Or K = 9) And (l = 4 Or l = 5 Or l = 6) Then
    Wert4 = Worksheets("Spielfelder").Cells(10 + K, 1 + l)
Elseif (i = 7 Or i = 8 Or i = 9) And (j = 7 Or j = 8 Or j = 9) And _
    (K = 7 Or K = 8 Or K = 9) And (l = 7 Or l = 8 Or l = 9) Then
    Wert4 = Worksheets("Spielfelder").Cells(10 + K, 1 + l)
End If
Next l

If (j = l) And (K = i) Then
    l = l + 1
Else
    If (Wert1 = Wert4) And (Not (Wert1 = 0)) Then
        MsgBox "Fehlerhafte Hypothese " & "x= " & j & "; y= " & i & "; Wert= " & Wert1
        l = 9
        K = 9
        j = 9
        i = 9
        Stopp = 1
    End If
End If
Next K
Next j
Next i
PlausibilitätsprüfungBloecke = Stopp
End Function

```

## Function HypothesePruefen()

*'UserForm1>Hypothesen>Hypothesen überprüfen>CommandButton14\_Click()*

*'Algorithmus: Addiere alle Werte der vier Bereiche und prüfe ob in allen Feldern eine Neun steht,*

*'was bedeutet, dass das Sudoku fertig gelöst ist. Danach geht eine MsgBox auf.*

```

Dim Anz As Integer
Anz = 0
Dim r1, r2, r3, r4 As Range
Set r1 = Worksheets("Tabellen").Range("N11:V19")
Set r2 = Worksheets("Tabellen").Range("N21:V29")
Set r3 = Worksheets("Tabellen").Range("N31:V39")
Set r4 = Worksheets("Tabellen").Range("N41:V49")

```

```

For i = 1 To 9
    For j = 1 To 9
        Anz = Anz + r1(i, j)
    Next j
Next i

```

```

For i = 1 To 9
    For j = 1 To 9

```

```

        Anz = Anz + r2(i, j)
    Next j
Next i

For i = 1 To 9
    For j = 1 To 9
        Anz = Anz + r3(i, j)
    Next j
Next i

For i = 1 To 9
    For j = 1 To 9
        Anz = Anz + r4(i, j)
    Next j
Next i

If Anz = (4 * 9 * 9 * 9) Then
    MsgBox "Gratulation! Die Hypothese ist korrekt. Sie haben das Sudoku erfolgreich gelöst."
Else
    MsgBox "Schade! Sie haben das Sudoku noch nicht gelöst. Entweder war die Hypothese falsch, ungenügend oder die Anzahl Iterationen zu klein."
End If

```

## **End Function**

### **Private Sub CommandButton16\_Click()**

```

'UserForm1>Hypothesen>Hypothese widerrufen
'Algorithmus: Ruft die zwei Funktionen auf
    Call HypotheseprüfungRückgängig
    Call HypotheseRückgängig

```

## **End Sub**

### **Function HypotheseprüfungRückgängig()**

```

'UserForm1>Hypothesen>Hypothese widerrufen>CommandButton16_Click()
'Algorithmus: Lösche im Tabellenblatt "Spielfelder" alle Felder bei der die Eigenschaft
Font.Color
'einen der RGB Werte annimmt.

```

```

    For i = 1 To 9
        For j = 1 To 9
            If Worksheets("Spielfelder").Cells(10 + i, 1 + j).Font.Color = _
                (RGB(255, 0, 0) Or RGB(254, 0, 0) Or _
                RGB(253, 0, 0) Or RGB(252, 0, 0)) Then
                Worksheets("Spielfelder").Cells(10 + i, 1 + j).ClearContents
            End If
        Next j
    Next i

```

## **End Function**

## Function HypotheseRückgängig()

*'UserForm1>Hypothesen>Hypothese widerrufen>CommandButton16\_Click()  
'Algorithmus:Lösche im Tabellenblatt "Spielfelder" die Felder bei der die Eigenschaft  
Font.Color*

*'den RGB Werte annimmt(die Hypothesefelder).*

```
For i = 1 To 9
    For j = 1 To 9
        If Worksheets("Spielfelder").Cells(10 + i, 1 + j).Font.Color = _
            (RGB(255, 0, 255)) Then
            Worksheets("Spielfelder").Cells(10 + i, 1 + j).ClearContents
        End If
    Next j
Next i
```

**End Function**

## File HedgeFunds-PolynomialGoalProgramming

```
Private Sub CommandButton1_Click()
    Dim MyLoadAreaR As Range, MyLoadArea As String
    Set MyLoadAreaR = Application.InputBox(Prompt:="Select MyLoadArea", Default:="=G20:G33",
    Type:=8)
    MyLoadArea = MyLoadAreaR.Address(RowAbsolute:=False, ColumnAbsolute:=False)
    Call Initialisation
    Call MySolver(MyLoadArea)
End Sub
```

```
Function Initialisation()
    Worksheets("Tabelle1").Cells(30, 2) = 0.5
    Worksheets("Tabelle1").Cells(30, 3) = 0.5
    Worksheets("Tabelle1").Cells(24, 2) = 0
    Worksheets("Tabelle1").Cells(24, 3) = 0
    Worksheets("Tabelle1").Cells(24, 4) = 0
    Worksheets("Tabelle1").Cells(24, 5) = 0
End Function
```

```
Function MySolver(MyLoadArea As String)
    'Solver starten
    Worksheets("Tabelle1").Activate
    SolverReset
    SolverLoad loadArea:=Range(MyLoadArea)
    SolverSolve (True)
End Function
```

```
Private Sub CommandButton2_Click()
    Dim MyLoadAreaR As Range, MyLoadArea As String
    Set MyLoadAreaR = Application.InputBox(Prompt:="Select MyLoadArea", Default:="=B43:B47",
    Type:=8)
    MyLoadArea = MyLoadAreaR.Address(RowAbsolute:=False, ColumnAbsolute:=False)
    Call MyReturnsFct(MyLoadArea)
End Sub
```

```

Function MyReturnsFct(MyLoadArea As String)
Dim MyReturns(5, 2) As Double, MyWeightsT(1, 2) As Double, MyPortfolioReturns(5) As Double
For j = 2 To 3
For i = 3 To 7
    MyReturns(i - 2, j - 1) = Worksheets("Tabelle1").Cells(i, j)
Next i
Next j
For j = 2 To 3
    MyWeightsT(1, j - 1) = Worksheets("Tabelle1").Cells(30, j)
Next j

For i = 53 To 57
    MyPortfolioReturns(i - 52) = (MyReturns(i - 52, 1) * MyWeightsT(1, 1) + MyReturns(i - 52, 2) *
MyWeightsT(1, 2))
Next i
For i = 1 To 5
    Worksheets("Tabelle1").Range(MyLoadArea).Cells(i, 1) = MyPortfolioReturns(i)
Next i
End Function

```

## **File: HedgeFunds – UtilityOptimization**

```

Private Sub CommandButton1_Click()
MyLoadArea = "A77:N77"
a = 0.1
saveline = 82

For i = 1 To 10
Worksheets("MV-UtilityMaximization").Cells(70, 2).Value = a
'Solver starten
Worksheets("MV-UtilityMaximization").Activate
SolverReset
SolverLoad loadArea:=Range(MyLoadArea)
SolverSolve (True)
'Ergebnisse speichern
    Worksheets("MV-UtilityMaximization").Cells(saveline, 1).Value = _
        Worksheets("MV-UtilityMaximization").Cells(70, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 2).Value = _
        Worksheets("MV-UtilityMaximization").Cells(69, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 3).Value = _
        Worksheets("MV-UtilityMaximization").Cells(68, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 4).Value = _
        Worksheets("MV-UtilityMaximization").Cells(67, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 5).Value = _
        Worksheets("MV-UtilityMaximization").Cells(58, 1).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 6).Value = _
        Worksheets("MV-UtilityMaximization").Cells(59, 1).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 7).Value = _
        Worksheets("MV-UtilityMaximization").Cells(60, 1).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 8).Value = _
        Worksheets("MV-UtilityMaximization").Cells(61, 1).Value

```

```

Worksheets("MV-UtilityMaximization").Cells(saveline, 9).Value = _
    Worksheets("MV-UtilityMaximization").Cells(62, 1).Value
saveline = saveline + 1
a = a + 2
Next i
End Sub

```

```

Private Sub CommandButton2_Click()
MyLoadArea = "A107:N107"
Dim a(3), b(3), c(3) As Double
a(1) = (0.1)
a(2) = (2)
a(3) = 15
b(1) = (0.1)
b(2) = (2)
b(3) = 15
c(1) = (0.1)
c(2) = (2)
c(3) = 15
saveline = 112
For i = 1 To 3
For j = 1 To 3
For k = 1 To 3
Worksheets("MV-UtilityMaximization").Cells(98, 2).Value = a(i)
Worksheets("MV-UtilityMaximization").Cells(99, 2).Value = a(j)
Worksheets("MV-UtilityMaximization").Cells(100, 2).Value = a(k)
'Solver starten
Worksheets("MV-UtilityMaximization").Activate
SolverReset
SolverLoad loadArea:=Range(MyLoadArea)
SolverSolve (True)
'Ergebnisse speichern
    Worksheets("MV-UtilityMaximization").Cells(saveline, 1).Value = _
        Worksheets("MV-UtilityMaximization").Cells(98, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 2).Value = _
        Worksheets("MV-UtilityMaximization").Cells(99, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 3).Value = _
        Worksheets("MV-UtilityMaximization").Cells(100, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 4).Value = _
        Worksheets("MV-UtilityMaximization").Cells(97, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 5).Value = _
        Worksheets("MV-UtilityMaximization").Cells(68, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 6).Value = _
        Worksheets("MV-UtilityMaximization").Cells(67, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 7).Value = _
        Worksheets("MV-UtilityMaximization").Cells(95, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 8).Value = _
        Worksheets("MV-UtilityMaximization").Cells(96, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 9).Value = _
        Worksheets("MV-UtilityMaximization").Cells(58, 1).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 10).Value = _
        Worksheets("MV-UtilityMaximization").Cells(59, 1).Value

```

```

Worksheets("MV-UtilityMaximization").Cells(saveline, 11).Value = _
    Worksheets("MV-UtilityMaximization").Cells(60, 1).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 12).Value = _
    Worksheets("MV-UtilityMaximization").Cells(61, 1).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 13).Value = _
    Worksheets("MV-UtilityMaximization").Cells(62, 1).Value
saveline = saveline + 1
Next k
Next j
Next i
End Sub

```

## File: MarkowitzPortfolioOptimization- InternationalDiversification

```

Private Sub CommandButton1_Click()
MyLoadArea = "A77:N77"
a = 0.1
saveline = 82

For i = 1 To 10
Worksheets("MV-UtilityMaximization").Cells(70, 2).Value = a
'Solver starten
Worksheets("MV-UtilityMaximization").Activate
SolverReset
SolverLoad loadArea:=Range(MyLoadArea)
SolverSolve (True)
'Ergebnisse speichern
    Worksheets("MV-UtilityMaximization").Cells(saveline, 1).Value = _
        Worksheets("MV-UtilityMaximization").Cells(70, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 2).Value = _
        Worksheets("MV-UtilityMaximization").Cells(69, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 3).Value = _
        Worksheets("MV-UtilityMaximization").Cells(68, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 4).Value = _
        Worksheets("MV-UtilityMaximization").Cells(67, 2).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 5).Value = _
        Worksheets("MV-UtilityMaximization").Cells(58, 1).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 6).Value = _
        Worksheets("MV-UtilityMaximization").Cells(59, 1).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 7).Value = _
        Worksheets("MV-UtilityMaximization").Cells(60, 1).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 8).Value = _
        Worksheets("MV-UtilityMaximization").Cells(61, 1).Value
    Worksheets("MV-UtilityMaximization").Cells(saveline, 9).Value = _
        Worksheets("MV-UtilityMaximization").Cells(62, 1).Value
    saveline = saveline + 1
    a = a + 2
Next i
End Sub

```



```

Private Sub CommandButton2_Click()
MyLoadArea = "A107:N107"
Dim a(3), b(3), c(3) As Double
a(1) = (0.1)
a(2) = (2)
a(3) = 15
b(1) = (0.1)
b(2) = (2)
b(3) = 15
c(1) = (0.1)
c(2) = (2)
c(3) = 15
saveline = 112
For i = 1 To 3
For j = 1 To 3
For k = 1 To 3
Worksheets("MV-UtilityMaximization").Cells(98, 2).Value = a(i)
Worksheets("MV-UtilityMaximization").Cells(99, 2).Value = a(j)
Worksheets("MV-UtilityMaximization").Cells(100, 2).Value = a(k)
'Solver starten
Worksheets("MV-UtilityMaximization").Activate
SolverReset
SolverLoad loadArea:=Range(MyLoadArea)
SolverSolve (True)
'Ergebnisse speichern
Worksheets("MV-UtilityMaximization").Cells(saveline, 1).Value = _
    Worksheets("MV-UtilityMaximization").Cells(98, 2).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 2).Value = _
    Worksheets("MV-UtilityMaximization").Cells(99, 2).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 3).Value = _
    Worksheets("MV-UtilityMaximization").Cells(100, 2).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 4).Value = _
    Worksheets("MV-UtilityMaximization").Cells(97, 2).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 5).Value = _
    Worksheets("MV-UtilityMaximization").Cells(68, 2).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 6).Value = _
    Worksheets("MV-UtilityMaximization").Cells(67, 2).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 7).Value = _
    Worksheets("MV-UtilityMaximization").Cells(95, 2).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 8).Value = _
    Worksheets("MV-UtilityMaximization").Cells(96, 2).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 9).Value = _
    Worksheets("MV-UtilityMaximization").Cells(58, 1).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 10).Value = _
    Worksheets("MV-UtilityMaximization").Cells(59, 1).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 11).Value = _
    Worksheets("MV-UtilityMaximization").Cells(60, 1).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 12).Value = _
    Worksheets("MV-UtilityMaximization").Cells(61, 1).Value
Worksheets("MV-UtilityMaximization").Cells(saveline, 13).Value = _
    Worksheets("MV-UtilityMaximization").Cells(62, 1).Value

```

```

        saveline = saveline + 1
Next k
Next j
Next i
End Sub

Private Sub CommandButton1_Click()
MyLoadArea = "A77:L77"
a = 0
saveline = 82

For i = 1 To 10
Worksheets("UtilityMaximization").Cells(70, 2).Value = a
'Solver starten
Worksheets("UtilityMaximization").Activate
SolverReset
SolverLoad loadArea:=Range(MyLoadArea)
SolverSolve (True)
'Ergebnisse speichern
    Worksheets("UtilityMaximization").Cells(saveline, 1).Value = _
        Worksheets("UtilityMaximization").Cells(70, 2).Value
    Worksheets("UtilityMaximization").Cells(saveline, 2).Value = _
        Worksheets("UtilityMaximization").Cells(69, 2).Value
    Worksheets("UtilityMaximization").Cells(saveline, 3).Value = _
        Worksheets("UtilityMaximization").Cells(68, 2).Value
    Worksheets("UtilityMaximization").Cells(saveline, 4).Value = _
        Worksheets("UtilityMaximization").Cells(67, 2).Value
    Worksheets("UtilityMaximization").Cells(saveline, 5).Value = _
        Worksheets("UtilityMaximization").Cells(58, 1).Value
    Worksheets("UtilityMaximization").Cells(saveline, 6).Value = _
        Worksheets("UtilityMaximization").Cells(59, 1).Value
    Worksheets("UtilityMaximization").Cells(saveline, 7).Value = _
        Worksheets("UtilityMaximization").Cells(60, 1).Value
    Worksheets("UtilityMaximization").Cells(saveline, 8).Value = _
        Worksheets("UtilityMaximization").Cells(61, 1).Value
    Worksheets("UtilityMaximization").Cells(saveline, 9).Value = _
        Worksheets("UtilityMaximization").Cells(62, 1).Value
    Worksheets("UtilityMaximization").Cells(saveline, 10).Value = _
        Worksheets("UtilityMaximization").Cells(63, 1).Value
    Worksheets("UtilityMaximization").Cells(saveline, 11).Value = _
        Worksheets("UtilityMaximization").Cells(64, 1).Value
    saveline = saveline + 1
    a = a + 1
Next i
End Sub

```

## **File: OptionHedging – BlackScholes**

```

Private Sub CommandButton1_Click()

```

```

Dim MyLoadAreaR As Range, MyLoadArea As String
Set MyLoadAreaR = Application.InputBox(Prompt:="Select MyLoadArea", Default:="H4:H11",
Type:=8)
MyLoadArea = MyLoadAreaR.Address(RowAbsolute:=False, ColumnAbsolute:=False)
Call MySolver(MyLoadArea)
End Sub

```

```

Function MySolver(MyLoadArea As String)
'Solver starten
Worksheets("B-S Mispriced Option").Activate
SolverReset
SolverLoad loadArea:=Range(MyLoadArea)
SolverSolve (True)
'balloon
Set MyBalloon = Assistant.NewBalloon
With MyBalloon
.HHeading = "Option Type"
.Text = "Select your option type"
.CheckBoxes(1).Text = "Put Option"
.CheckBoxes(2).Text = "Call Option"
.Button = msoButtonSetOK
.Show
If .CheckBoxes(1).Checked Then
    MyAnswer = 1
End If
If .CheckBoxes(2).Checked Then
    MyAnswer = 2
End If
End With
'load area setzieren
MySplit = Strings.Split(MyLoadArea, ":")
MyLine = Strings.StrReverse(MySplit(0))
MyLine = Conversion.Val(MyLine)
MyLine = Strings.StrReverse(MyLine)
MySaveArea = "F" & MyLine
'Ergebnisse links nach rechts übertragen
If MyAnswer = 1 Then
    Worksheets("B-S Mispriced Option").Range(MySaveArea).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(19, 2).Value
Elseif MyAnswer = 2 Then
    Worksheets("B-S Mispriced Option").Range(MySaveArea).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(18, 2).Value
End If
    ActiveCell.Offset(2, 0).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(2, 2).Value
    ActiveCell.Offset(1, 0).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(3, 2).Value
    ActiveCell.Offset(1, 0).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(4, 2).Value
    ActiveCell.Offset(1, 0).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(10, 2).Value
    ActiveCell.Offset(1, 0).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(6, 2).Value

```

End Function

```
Private Sub CommandButton2_Click()  
    Dim MyLoadAreaR As Range  
    Dim MyLoadArea, MySRange, MyPRange As String  
    Dim MySValue, MyPValue As Double  
    'input  
    Set MyLoadAreaR = Application.InputBox(Prompt:="Select MyLoadArea", Default:="", Left:=80, Top:=250, Type:=8)  
    MyLoadArea = MyLoadAreaR.Address(RowAbsolute:=False, ColumnAbsolute:=False)  
    'load area setzieren  
    'Bem.: Andere Möglichkeit mit Worksheets.Range.Activate und ActiveCell.Offset.Activate wie oben  
    MyFirstFieldA1 = Strings.Split(MyLoadArea, ":")  
    MyFirstFieldR1C1 = Application.ConvertFormula(MyFirstFieldA1, xlA1, xlR1C1, xlAbsolute)  
    MyFirstFieldSplitR1C1 = Strings.Split(MyFirstFieldR1C1(1), "C")  
    MyFirstFieldSplitR1 = Strings.Split(MyFirstFieldSplitR1C1(0), "R")  
    MyFirstFieldR = MyFirstFieldSplitR1(1)  
    MyFirstFieldC = MyFirstFieldSplitR1C1(1)  
    MyTRange = Application.ConvertFormula("R" & CStr(MyFirstFieldR + 0) & "C" &  
CStr(MyFirstFieldC), xlR1C1, xlA1)  
    MySRange = Application.ConvertFormula("R" & CStr(MyFirstFieldR + 1) & "C" &  
CStr(MyFirstFieldC), xlR1C1, xlA1)  
    MyPRange = Application.ConvertFormula("R" & CStr(MyFirstFieldR + 2) & "C" &  
CStr(MyFirstFieldC), xlR1C1, xlA1)  
    MyHRange = Application.ConvertFormula("R" & CStr(MyFirstFieldR + 3) & "C" &  
CStr(MyFirstFieldC), xlR1C1, xlA1)  
    'Std Wert holen  
    MySValue = Worksheets("B-S Mispriced Option").Range(MyTRange).Value  
    'Std Wert setzen  
    Worksheets("B-S Mispriced Option").Range("B6").Value = MySValue  
    'S Wert holen  
    MySValue = Worksheets("B-S Mispriced Option").Range(MySRange).Value  
    'S Wert setzen  
    Worksheets("B-S Mispriced Option").Range("B5").Value = MySValue  
    'P Wert holen  
    MyPValue = Worksheets("B-S Mispriced Option").Range("B19").Value  
    'P Wert setzen  
    Worksheets("B-S Mispriced Option").Range(MyPRange).Value = MyPValue  
    'H Wert holen  
    MyHValue = Worksheets("B-S Mispriced Option").Range("B24").Value  
    'QP Wert holen  
    MyQPValue = Worksheets("B-S Mispriced Option").Range("E40").Value  
    'QS Wert berechnen  
    MyQSValue = WorksheetFunction.Round((MyQPValue * (-1) * MyHValue), 0)  
    'QS Wert setzen  
    Worksheets("B-S Mispriced Option").Range(MyHRange).Value = MyQSValue  
End Sub
```

```
Private Sub CommandButton1_Click()  
    Dim MyLoadAreaR As Range, MyLoadArea, MySplit, MyReferenceRange As String  
    'input  
    Set MyLoadAreaR = Application.InputBox(Prompt:="Select MyLoadArea", Default:="", Left:=530, Top:=50, Type:=8)
```

```

MyLoadArea = MyLoadAreaR.Address(RowAbsolute:=False, ColumnAbsolute:=False)
'get the first Range from My Load Area
MySplit = Strings.Split(MyLoadArea, ":")
MyReferenceRange = MySplit(0)
'call Black Scholes
Call MyBS(MyReferenceRange)
End Sub

```

```

Function MyBS(MyReferenceRange As String)
'load values
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
ActiveCell.Offset(0, 0).Activate
MyTValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
ActiveCell.Offset(1, 0).Activate
MyrValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
ActiveCell.Offset(2, 0).Activate
MyXValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
ActiveCell.Offset(3, 0).Activate
MySValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
ActiveCell.Offset(4, 0).Activate
MyStdValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
ActiveCell.Offset(5, 0).Activate
MyqValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
ActiveCell.Offset(6, 0).Activate
MyTqValue = ActiveCell.Value
'result area
MyTRange_ = "B2"
MyrRange_ = "B3"
MyXRange_ = "B4"
MySRange_ = "B5"
MyStdRange_ = "B6"
MyqRange_ = "B8"
MyTqRange_ = "B7"
MyPRange_ = "B19"
MyCRange_ = "B18"
MyHPRange_ = "B24"
MyHCRRange_ = "B23"
'Set Values in the result area
Worksheets("B-S Mispriced Option2").Range(MyTRange_).Value = MyTValue
Worksheets("B-S Mispriced Option2").Range(MyrRange_).Value = MyrValue
Worksheets("B-S Mispriced Option2").Range(MyXRange_).Value = MyXValue
Worksheets("B-S Mispriced Option2").Range(MySRange_).Value = MySValue
Worksheets("B-S Mispriced Option2").Range(MyStdRange_).Value = MyStdValue
Worksheets("B-S Mispriced Option2").Range(MyqRange_).Value = MyqValue
Worksheets("B-S Mispriced Option2").Range(MyTqRange_).Value = MyTqValue
'get Values in the result area
MyPValue = Worksheets("B-S Mispriced Option2").Range(MyPRange_).Value

```

```

MyCValue = Worksheets("B-S Mispriced Option2").Range(MyCRange_).Value
MyHPValue = Worksheets("B-S Mispriced Option2").Range(MyHPRange_).Value
MyHCValue = Worksheets("B-S Mispriced Option2").Range(MyHCRange_).Value
'Set Values in the load area
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(7, 0).Activate
    ActiveCell.Value = MyPValue
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(8, 0).Activate
    ActiveCell.Value = MyCValue
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(9, 0).Activate
    ActiveCell.Value = MyHPValue
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(10, 0).Activate
    ActiveCell.Value = MyHCValue

```

```

'Further Calculations in the load area
'MyHValue = Worksheets("B-S Mispriced Option").Range("B24").Value
'MyQPValue = Worksheets("B-S Mispriced Option").Range("E40").Value
'MyQSValue = WorksheetFunction.Round((MyQPValue * (-1) * MyHValue), 0)
'Worksheets("B-S Mispriced Option").Range(MyHRange).Value = MyQSValue

```

End Function

```

Private Sub CommandButton1_Click()
Dim MyLoadAreaR As Range, MyLoadArea, MySplit, MyReferenceRange As String
'input
Set MyLoadAreaR = Application.InputBox(Prompt:="Select MyLoadArea", Default:="=E3:E7",
Left:=530, Top:=50, Type:=8)
MyLoadArea = MyLoadAreaR.Address(RowAbsolute:=False, ColumnAbsolute:=False)
'get the first Range from My Load Area
MySplit = Strings.Split(MyLoadArea, ":")
MyReferenceRange = MySplit(0)
'call Black Scholes
Call MyBS(MyReferenceRange)
End Sub

```

```

Function MyBS(MyReferenceRange As String)
'load values
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(0, 0).Activate
    MyTValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(1, 0).Activate
    MyrValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(2, 0).Activate
    MyXValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(3, 0).Activate

```

```

    MySValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(4, 0).Activate
    MyStdValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(5, 0).Activate
    MyqValue = ActiveCell.Value
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(6, 0).Activate
    MyTqValue = ActiveCell.Value
'result area
MyTRange_ = "B2"
MyrRange_ = "B3"
MyXRange_ = "B4"
MySRange_ = "B5"
MyStdRange_ = "B6"
MyqRange_ = "B8"
MyTqRange_ = "B7"
MyPRange_ = "B19"
MyCRange_ = "B18"
MyHPRange_ = "B24"
MyHCRRange_ = "B23"
'Set Values in the result area
Worksheets("B-S Mispriced Option2").Range(MyTRange_).Value = MyTValue
Worksheets("B-S Mispriced Option2").Range(MyrRange_).Value = MyrValue
Worksheets("B-S Mispriced Option2").Range(MyXRange_).Value = MyXValue
Worksheets("B-S Mispriced Option2").Range(MySRange_).Value = MySValue
Worksheets("B-S Mispriced Option2").Range(MyStdRange_).Value = MyStdValue
Worksheets("B-S Mispriced Option2").Range(MyqRange_).Value = MyqValue
Worksheets("B-S Mispriced Option2").Range(MyTqRange_).Value = MyTqValue
'get Values in the result area
MyPValue = Worksheets("B-S Mispriced Option2").Range(MyPRange_).Value
MyCValue = Worksheets("B-S Mispriced Option2").Range(MyCRange_).Value
MyHPValue = Worksheets("B-S Mispriced Option2").Range(MyHPRange_).Value
MyHCValue = Worksheets("B-S Mispriced Option2").Range(MyHCRRange_).Value
'Set Values in the load area
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(7, 0).Activate
    ActiveCell.Value = MyPValue
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(8, 0).Activate
    ActiveCell.Value = MyCValue
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(9, 0).Activate
    ActiveCell.Value = MyHPValue
Worksheets("B-S Mispriced Option2").Range(MyReferenceRange).Activate
    ActiveCell.Offset(10, 0).Activate
    ActiveCell.Value = MyHCValue
End Function

Private Sub CommandButton2_Click()
'Call MyBalloon to choose strategy
Dim MyAnswer As Integer

```

```

Call MyBalloonHedgingStrategies(MyAnswer)
If MyAnswer = 1 Then
    Call Delta0CrossHedCallCall
ElseIf MyAnswer = 2 Then
    'Call Delta0CrossHedPutPut
ElseIf MyAnswer = 3 Then
    'Call Delta0CallStock
ElseIf MyAnswer = 4 Then
    'Call Delta0PutStock
End If
End Sub
Function MyBalloonHedgingStrategies(MyAnswer As Integer)
'balloon
    Set MyBalloon = Assistant.NewBalloon
    With MyBalloon
        .Heading = "Option Type"
        .Text = "Select your option type"
        .CheckBoxes(1).Text = "Strategy: Delta-Neutral Cross Hedging with two Call Options"
        .CheckBoxes(2).Text = "Strategy: Delta-Neutral Cross Hedging with two Put Options"
        .CheckBoxes(3).Text = "Strategy: Delta-Neutral with Call Option and Underlying Stock"
        .CheckBoxes(4).Text = "Strategy: Delta-Neutral with Put Option and Underlying Stock"
        .Button = msoButtonSetOkCancel
        .Show
        If .CheckBoxes(1).Checked Then
            MyAnswer = 1
        End If
        If .CheckBoxes(2).Checked Then
            MyAnswer = 2
        End If
        If .CheckBoxes(3).Checked Then
            MyAnswer = 3
        End If
        If .CheckBoxes(4).Checked Then
            MyAnswer = 4
        End If
    End With
    MyBalloon = MyAnswer
End Function

Function Delta0CrossHedCallCall()
'1 Clear
Worksheets("B-S Mispriced Option3").Range("H14:I14").ClearContents
Worksheets("B-S Mispriced Option3").Range("E18:F22").ClearContents

'1 Hedge Ration
MyHC1Value = Worksheets("B-S Mispriced Option3").Range("H13").Value
MyHC2Value = Worksheets("B-S Mispriced Option3").Range("I13").Value

MyHedgeValue = MyHC1Value / MyHC2Value
If MyHedgeValue >= 1 Then
    MyCase = 1
End If

```



```

If MyCase = 1 Then
    Worksheets("B-S Mispriced Option3").Range("I14").Formula = "=H13/I13"
Else
    Worksheets("B-S Mispriced Option3").Range("H14").Formula = "=I13/H13"
End If
'2 Nof options
If MyCase = 1 Then
    MsgBox ("Check the value of the range F16. It should be negative")
    Worksheets("B-S Mispriced Option3").Range("F19").Formula = "=I14*E16*(-1)"
Else
    MsgBox ("Check the value of the range F16. It should be negative")
    Worksheets("B-S Mispriced Option3").Range("E19").Formula = "=E14*F16*(-1)"
End If
'3 Initial Cash Flow
If MyCase = 1 Then
    Worksheets("B-S Mispriced Option3").Range("E21").Formula = "=E11*E16"
    Worksheets("B-S Mispriced Option3").Range("F21").Formula = "=F11*F19"
    Worksheets("B-S Mispriced Option3").Range("E22").Formula = "=E21+F21"
Else
    Worksheets("B-S Mispriced Option3").Range("E21").Formula = "=E11*E19"
    Worksheets("B-S Mispriced Option3").Range("F21").Formula = "=F11*F16"
    Worksheets("B-S Mispriced Option3").Range("F22").Formula = "=E21+F21"
End If

End Function
Function Delta0CrossHedPutPut()

End Function
Function Delta0CallStock()

End Function
Function Delta0CallPut()

End Function

Private Sub CommandButton3_Click()
'Call MyBalloon to choose strategy
Dim MyAnswer As Integer
Call MyBalloonHedgingStrategies(MyAnswer)
If MyAnswer = 1 Then
    Call Delta0CrossHedCallCallScenario
ElseIf MyAnswer = 2 Then
    'Call Delta0CrossHedPutPut
ElseIf MyAnswer = 3 Then
    'Call Delta0CallStock
ElseIf MyAnswer = 4 Then
    'Call Delta0PutStock
End If

End Sub

Function Delta0CrossHedCallCallScenario()

```

```


Set MyLoadAreaR = Application.InputBox(Prompt:="Select MyLoadArea", Default:="=E36:F45",
Left:=530, Top:=50, Type:=8)
MyLoadArea = MyLoadAreaR.Address(RowAbsolute:=False, ColumnAbsolute:=False)
'get the first Range from My Load Area
MySplit = Strings.Split(MyLoadArea, ":")
MyReferenceRange = MySplit(0)
'Clear
Worksheets("B-S Mispriced Option3").Range(MyLoadArea).ClearContents
'Hedge Ration
Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
ActiveCell.Offset(-1, 0).Activate
MyHC1Value = ActiveCell.Value
ActiveCell.Offset(0, 1).Activate
MyHC2Value = ActiveCell.Value

MyHedgeValue = MyHC1Value / MyHC2Value
If MyHedgeValue >= 1 Then
    MyCase = 1
End If

If MyCase = 1 Then
    Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
    MyL35 = ActiveCell.Offset(-1, 0).Address
    MyR35 = ActiveCell.Offset(-1, 1).Address
    Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
    ActiveCell.Offset(0, 0).Activate
    ActiveCell.Formula = "=" & MyL35 & "/" & MyR35
Else
    Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
    MyL35 = ActiveCell.Offset(-1, 1).Address
    MyR35 = ActiveCell.Offset(-1, 0).Address
    ActiveCell.Offset(0, 0).Activate
    ActiveCell.Formula = "=" & MyR35 & "/" & MyL35
End If
'Nof options
If MyCase = 1 Then
    Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
    ActiveCell.Offset(2, 0).Activate
    ActiveCell.Formula = "=-E16"
    Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
    ActiveCell.Offset(5, 1).Activate
    ActiveCell.Formula = "=-F19"
Else
    Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
    ActiveCell.Offset(2, 1).Activate
    ActiveCell.Formula = "=-F16"
    Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
    ActiveCell.Offset(5, 0).Activate
    ActiveCell.Formula = "=-E19"
End If
'Cash Flow
Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate

```

```

MyR33 = ActiveCell.Offset(-3, 1).Address
MyR36 = ActiveCell.Offset(0, 1).Address
MyL33 = ActiveCell.Offset(-3, 0).Address
MyL38 = ActiveCell.Offset(2, 0).Address
MyR38 = ActiveCell.Offset(2, 1).Address
MyL41 = ActiveCell.Offset(5, 0).Address
MyR41 = ActiveCell.Offset(5, 1).Address
MyR43 = ActiveCell.Offset(7, 1).Address
MyL43 = ActiveCell.Offset(7, 0).Address
MyL44 = ActiveCell.Offset(8, 0).Address
MyL22 = ActiveCell.Offset(-14, 0).Address

```

```

If MyCase = 1 Then
    Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
    ActiveCell.Offset(7, 0).Activate
        ActiveCell.Formula = "=" & MyL33 & "*" & MyL38
    ActiveCell.Offset(0, 1).Activate
        ActiveCell.Formula = "=" & MyR33 & "*" & MyR41
    ActiveCell.Offset(1, -1).Activate
        ActiveCell.Formula = "=" & MyL43 & "+" & MyR43
    ActiveCell.Offset(1, 0).Activate
        ActiveCell.Formula = "=" & MyL44 & "+" & "E22"
Else
    Worksheets("B-S Mispriced Option3").Range(MyReferenceRange).Activate
    ActiveCell.Offset(7, 0).Activate
        ActiveCell.Formula = "=" & MyL33 & "*" & MyL41
    ActiveCell.Offset(0, 1).Activate
        ActiveCell.Formula = "=" & MyR33 & "*" & MyR38
    ActiveCell.Offset(1, 0).Activate
        ActiveCell.Formula = "=" & MyL43 & "+" & MyR43
    ActiveCell.Offset(1, 0).Activate
        ActiveCell.Formula = "=" & MyL44 & "+" & "E22"
End If
End Function

```

```

Sub te()
A = "=" & "23" & "/" & "*"
End Sub

```

## **File: OptionPricing -BinomialModel -AmericanOptions**

```

Private Sub CommandButton1_Click()
    UserForm1.Left = 430
    UserForm1.Top = 300
    UserForm1.TextBox1.SelStart = 0
    UserForm1.TextBox1.SelLength = UserForm1.TextBox1.TextLength
    UserForm1.TextBox2.SelStart = 0
    UserForm1.TextBox2.SelLength = UserForm1.TextBox1.TextLength
    UserForm1.TextBox3.SelStart = 0
    UserForm1.TextBox3.SelLength = UserForm1.TextBox1.TextLength

```

```

UserForm1.TextBox4.SelStart = 0
UserForm1.TextBox4.SelLength = UserForm1.TextBox1.TextLength
UserForm1.TextBox5.SelStart = 0
UserForm1.TextBox5.SelLength = UserForm1.TextBox1.TextLength
UserForm1.TextBox6.SelStart = 0
UserForm1.TextBox6.SelLength = UserForm1.TextBox1.TextLength
UserForm1.TextBox7.SelStart = 0
UserForm1.TextBox7.SelLength = UserForm1.TextBox1.TextLength
UserForm1.TextBox8.SelStart = 0
UserForm1.TextBox8.SelLength = UserForm1.TextBox1.TextLength
UserForm1.TextBox9.SelStart = 0
UserForm1.TextBox9.SelLength = UserForm1.TextBox1.TextLength
UserForm1.TextBox10.SelStart = 0
UserForm1.TextBox10.SelLength = UserForm1.TextBox1.TextLength
UserForm1.TextBox11.SelStart = 0
UserForm1.TextBox11.SelLength = UserForm1.TextBox1.TextLength
myInput = Application.InputBox("1-Stock, 2-Forex, 3-Futures", , "1", , , , 2)
If myInput = "1" Then
    UserForm1.TextBox1.Text = "In!$B$6"
    UserForm1.TextBox2.Text = "In!$B$5"
    UserForm1.TextBox3.Text = "In!$B$8"
    UserForm1.TextBox4.Text = "In!$B$9"
    UserForm1.TextBox5.Text = "In!$B$15"
    UserForm1.TextBox6.Text = "In!$E$5"
    UserForm1.TextBox7.Text = "In!$E$6"
    UserForm1.TextBox8.Text = "In!$E$7"
    UserForm1.TextBox9.Text = "In!$E$8"
    UserForm1.TextBox10.Text = "In!$E$9"
    UserForm1.TextBox11.Text = "In!$E$10"
ElseIf myInput = "2" Then
    UserForm1.TextBox1.Text = "In!$B$21"
    UserForm1.TextBox2.Text = "In!$B$20"
    UserForm1.TextBox3.Text = "In!$B$22"
    UserForm1.TextBox4.Text = "In!$B$26"
    UserForm1.TextBox5.Text = "In!$B$28"
    UserForm1.TextBox6.Text = "In!$E$20"
    UserForm1.TextBox7.Text = "In!$E$21"
    UserForm1.TextBox8.Text = "In!$E$22"
    UserForm1.TextBox9.Text = "In!$E$23"
    UserForm1.TextBox10.Text = "In!$E$24"
    UserForm1.TextBox11.Text = "In!$E$25"
ElseIf myInput = "3" Then
    UserForm1.TextBox1.Text = "In!$B$36"
    UserForm1.TextBox2.Text = "In!$B$35"
    UserForm1.TextBox3.Text = "In!$B$37"
    UserForm1.TextBox4.Text = "In!$B$39"
    UserForm1.TextBox5.Text = "In!$B$40"
    UserForm1.TextBox6.Text = "In!$E$35"
    UserForm1.TextBox7.Text = "In!$E$36"
    UserForm1.TextBox8.Text = "In!$E$37"
    UserForm1.TextBox9.Text = "In!$E$38"
    UserForm1.TextBox10.Text = "In!$E$39"
    UserForm1.TextBox11.Text = "In!$E$40"

```

```

End If
Worksheets("In").Activate
UserForm1.Show
End Sub

```

```

Private Sub CommandButton2_Click()
'UserForm1>Abbrechen
    UserForm1.Hide
End Sub

```

```

Private Sub CommandButton1_Click()
'UserForm1>Stock:EuropeanStyleCall
    UserForm1.Hide
    Call GenerateEquationsEuropeanStyleCall("Stock", "EuropeanCallOptionOnStock")
End Sub

```

```

Private Sub CommandButton6_Click()
'UserForm1>Forex:EuropeanStyleCall
    UserForm1.Hide
    Call GenerateEquationsEuropeanStyleCall("Forex", "EuropeanCallOptionOnForex")
End Sub

```

```

Private Sub CommandButton9_Click()
'UserForm1>Futures:EuropeanStyleCall
    UserForm1.Hide
    Call GenerateEquationsEuropeanStyleCall("Futures", "EuropeanCallOptionOnFutures")
End Sub

```

```

Function GenerateEquationsEuropeanStyleCall(Underlying As String, OptionType As String)
K = UserForm1.TextBox1.Value
T = UserForm1.TextBox2.Value
S0 = UserForm1.TextBox3.Value
Std = UserForm1.TextBox4.Value
r = UserForm1.TextBox5.Value
Dt = UserForm1.TextBox6.Value
Up = UserForm1.TextBox7.Value
Down = UserForm1.TextBox8.Value
Prob = UserForm1.TextBox9.Value
EinsMinusProb = UserForm1.TextBox10.Value
n = UserForm1.TextBox11.Value
IterMax = Worksheets("In").Range(n).Value
'Clear all
myInput1 = Application.InputBox("Generate New Stock/Forex/Futures Values as well?", , "N", , , , 2)
If myInput1 = "N" Then
Else
    With Worksheets(Underlying)
        .Range(.Cells(3, 1), .Cells(257, 254)).ClearContents
    End With
End If
With Worksheets(OptionType)
    .Range(.Cells(3, 1), .Cells(257, 254)).ClearContents
End With

```

```

'Definitionen
mal = "*"
Gleich = "="
Hoch = "^"
minus = "-"
plus = "+"
'Programm für Stock
If myInput1 = "N" Then
Else
    For IterUnten = 0 To IterMax
        Worksheets(Underlying).Cells(3 + IterUnten, 1).Value = IterUnten
        For IterRechts = 0 To IterUnten
            'Formel
            UpStr = CStr(IterUnten - IterRechts)
            DownStr = CStr(IterRechts)
            Formel1 = Gleich & S0 & mal
            Formel2 = Up & Hoch & UpStr & mal
            Formel3 = Down & Hoch & DownStr
            Worksheets(Underlying).Cells(3 + IterUnten, 2 + IterRechts).Value = Formel1 & Formel2 &
Formel3
        Next IterRechts
    Next IterUnten
End If
'Programm für Call Option
'Formel für cT
For IterRechts = 0 To IterMax
    Formel4 = "R" & CStr(3 + IterMax) & "C" & CStr(2 + IterRechts)
    Formel5 = Application.ConvertFormula(Formula:=Formel4, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
    ST = Underlying & "!" & Formel5
    Formel6 = Gleich & "Max(" & ST & minus & K & ",0)"
    Worksheets(OptionType).Cells(3 + IterMax, 2 + IterRechts).Value = Formel6
Next IterRechts
'Formel für ct European Style
For IterRunter = 0 To IterMax - 1
    IterRauf = IterMax - IterRunter
    Worksheets(OptionType).Cells(3 + IterRauf, 1).Value = IterRauf
    For IterRechts = 0 To IterRauf - 1
        Formel7 = "R" & CStr(3 + IterRauf) & "C" & CStr(2 + IterRechts)
        ctj = Application.ConvertFormula(Formula:=Formel7, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
        Formel8 = "R" & CStr(3 + IterRauf) & "C" & CStr(2 + IterRechts + 1)
        ctjplus1 = Application.ConvertFormula(Formula:=Formel8, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
        Formel9 = Gleich & "exp(" & minus & r & mal & Dt & ")"
        Formel10 = mal & "(" & Prob & mal & ctj & plus
        Formel11 = EinsMinusProb & mal & ctjplus1 & ")"
        Formel12 = Formel9 & Formel10 & Formel11
        Worksheets(OptionType).Cells(3 + IterRauf - 1, 2 + IterRechts).Value = Formel12
    Next IterRechts
Next IterRunter
Worksheets(OptionType).Cells(3, 1).Value = 0
End Function

```

```

Private Sub CommandButton3_Click()
'UserForm1>Stock:AmericanStyleCall
    UserForm1.Hide
    Call GenerateEquationsAmericanStyleCall("Stock", "AmericanCallOptionOnStock")
End Sub
Private Sub CommandButton7_Click()
'UserForm1>Forex:AmericanStyleCall
    UserForm1.Hide
    Call GenerateEquationsAmericanStyleCall("Forex", "AmericanCallOptionOnForex")
End Sub
Private Sub CommandButton10_Click()
'UserForm1>Futures:AmericanStyleCall
    UserForm1.Hide
    Call GenerateEquationsAmericanStyleCall("Futures", "AmericanCallOptionOnFutures")
End Sub

Function GenerateEquationsAmericanStyleCall(Underlying As String, OptionType As String)
K = UserForm1.TextBox1.Value
T = UserForm1.TextBox2.Value
S0 = UserForm1.TextBox3.Value
Std = UserForm1.TextBox4.Value
r = UserForm1.TextBox5.Value
Dt = UserForm1.TextBox6.Value
Up = UserForm1.TextBox7.Value
Down = UserForm1.TextBox8.Value
Prob = UserForm1.TextBox9.Value
EinsMinusProb = UserForm1.TextBox10.Value
n = UserForm1.TextBox11.Value
IterMax = Worksheets("In").Range(n).Value
'Clear all
myInput1 = Application.InputBox("Generate New Stock/Forex/Futures Values as well?", , "N" , , , , 2)
If myInput1 = "N" Then
Else
    With Worksheets(Underlying)
        .Range(.Cells(3, 1), .Cells(257, 254)).ClearContents
    End With
End If
With Worksheets(OptionType)
    .Range(.Cells(3, 1), .Cells(257, 254)).ClearContents
End With
'Definitionen
mal = "*"
Gleich = "="
Hoch = "^"
minus = "-"
plus = "+"
'Programm für Stock
If myInput1 = "N" Then
Else
    For IterUnten = 0 To IterMax
        Worksheets(Underlying).Cells(3 + IterUnten, 1).Value = IterUnten
        For IterRechts = 0 To IterUnten

```

```

'Formel
UpStr = CStr(IterUnten - IterRechts)
DownStr = CStr(IterRechts)
Formel1 = Gleich & S0 & mal
Formel2 = Up & Hoch & UpStr & mal
Formel3 = Down & Hoch & DownStr
Worksheets(Underlying).Cells(3 + IterUnten, 2 + IterRechts).Value = Formel1 & Formel2 &
Formel3
Next IterRechts
Next IterUnten
End If
'Programm für Call Option
'Formel für cT
For IterRechts = 0 To IterMax
Formel4 = "R" & CStr(3 + IterMax) & "C" & CStr(2 + IterRechts)
Formel5 = Application.ConvertFormula(Formula:=Formel4, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
ST = Underlying & "!" & Formel5
Formel6 = Gleich & "Max(" & ST & minus & K & ",0)"
Worksheets(OptionType).Cells(3 + IterMax, 2 + IterRechts).Value = Formel6
Next IterRechts
'Formel für ct American Style
For IterRunter = 0 To IterMax - 1
IterRauf = IterMax - IterRunter
Worksheets(OptionType).Cells(3 + IterRauf, 1).Value = IterRauf
For IterRechts = 0 To IterRauf - 1
Formel7 = "R" & CStr(3 + IterRauf) & "C" & CStr(2 + IterRechts)
ctj = Application.ConvertFormula(Formula:=Formel7, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
Formel8 = "R" & CStr(3 + IterRauf) & "C" & CStr(2 + IterRechts + 1)
ctjplus1 = Application.ConvertFormula(Formula:=Formel8, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
Formel9 = "exp(" & minus & r & mal & Dt & ")"
Formel10 = mal & "(" & Prob & mal & ctj & plus
Formel11 = EinsMinusProb & mal & ctjplus1 & ")"
Formel12 = Formel9 & Formel10 & Formel11
Formel13 = "R" & CStr(3 + IterRauf - 1) & "C" & CStr(2 + IterRechts)
STminus1j = Application.ConvertFormula(Formula:=Formel13, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
Formel14 = Gleich & "Max(" & Underlying & "!" & STminus1j & minus & K
Formel15 = "," & Formel12 & ")"
Formel16 = Formel14 & Formel15
Worksheets(OptionType).Cells(3 + IterRauf - 1, 2 + IterRechts).Value = Formel16
Next IterRechts
Next IterRunter
Worksheets(OptionType).Cells(3, 1).Value = 0
End Function

Private Sub CommandButton4_Click()
'UserForm1>Stock:AmericanStylePut
UserForm1.Hide
Call GenerateEquationsAmericanStylePut("Stock", "AmericanPutOptionOnStock")
End Sub

```



```

Private Sub CommandButton8_Click()
'UserForm1>Forex:AmericanStylePut
    UserForm1.Hide
    Call GenerateEquationsAmericanStylePut("Forex", "AmericanPutOptionOnForex")
End Sub

```

```

Private Sub CommandButton11_Click()
'UserForm1>Futures:AmericanStylePut
    UserForm1.Hide
    Call GenerateEquationsAmericanStylePut("Futures", "AmericanPutOptionOnFutures")
End Sub

```

```

Sub GenerateEquationsAmericanStylePut(Underlying As String, OptionType As String)
K = UserForm1.TextBox1.Value
T = UserForm1.TextBox2.Value
S0 = UserForm1.TextBox3.Value
Std = UserForm1.TextBox4.Value
r = UserForm1.TextBox5.Value
Dt = UserForm1.TextBox6.Value
Up = UserForm1.TextBox7.Value
Down = UserForm1.TextBox8.Value
Prob = UserForm1.TextBox9.Value
EinsMinusProb = UserForm1.TextBox10.Value
n = UserForm1.TextBox11.Value
IterMax = Worksheets("In").Range(n).Value
'Clear all
myInput1 = Application.InputBox("Generate New Stock/Forex/Futures Values as well?", , "N", , , , 2)
If myInput1 = "N" Then
Else
    With Worksheets(Underlying)
        .Range(.Cells(3, 1), .Cells(257, 254)).ClearContents
    End With
End If
With Worksheets(OptionType)
    .Range(.Cells(3, 1), .Cells(257, 254)).ClearContents
End With
'Definitionen
mal = "*"
Gleich = "="
Hoch = "^"
minus = "-"
plus = "+"
'Programm für Stock
If myInput1 = "N" Then
Else
    For IterUnten = 0 To IterMax
        Worksheets(Underlying).Cells(3 + IterUnten, 1).Value = IterUnten
        For IterRechts = 0 To IterUnten
            'Formel
            UpStr = CStr(IterUnten - IterRechts)
            DownStr = CStr(IterRechts)
            Formel1 = Gleich & S0 & mal
            Formel2 = Up & Hoch & UpStr & mal

```

```

        Formel3 = Down & Hoch & DownStr
        Worksheets(Underlying).Cells(3 + IterUnten, 2 + IterRechts).Value = Formel1 & Formel2 &
Formel3
        Next IterRechts
    Next IterUnten
End If
'Programm für Put Option
'Formel für pT
For IterRechts = 0 To IterMax
    Formel4 = "R" & CStr(3 + IterMax) & "C" & CStr(2 + IterRechts)
    Formel5 = Application.ConvertFormula(Formula:=Formel4, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
    ST = Underlying & "!" & Formel5
    Formel6 = Gleich & "Max(" & K & minus & ST & ",0)"
    Worksheets(OptionType).Cells(3 + IterMax, 2 + IterRechts).Value = Formel6
Next IterRechts
'Formel für pt American Style
For IterRunter = 0 To IterMax - 1
    IterRauf = IterMax - IterRunter
    Worksheets(OptionType).Cells(3 + IterRauf, 1).Value = IterRauf
    For IterRechts = 0 To IterRauf - 1
        Formel7 = "R" & CStr(3 + IterRauf) & "C" & CStr(2 + IterRechts)
        ptj = Application.ConvertFormula(Formula:=Formel7, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
        Formel8 = "R" & CStr(3 + IterRauf) & "C" & CStr(2 + IterRechts + 1)
        ptjplus1 = Application.ConvertFormula(Formula:=Formel8, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
        Formel9 = "exp(" & minus & r & mal & Dt & ")"
        Formel10 = mal & "(" & Prob & mal & ptj & plus
        Formel11 = EinsMinusProb & mal & ptjplus1 & ")"
        Formel12 = Formel9 & Formel10 & Formel11
        Formel13 = "R" & CStr(3 + IterRauf - 1) & "C" & CStr(2 + IterRechts)
        STminus1j = Application.ConvertFormula(Formula:=Formel13, fromReferenceStyle:=xlR1C1,
toReferenceStyle:=xlA1)
        Formel14 = Gleich & "Max(" & K & minus & Underlying & "!" & STminus1j
        Formel15 = "," & Formel12 & ")"
        Formel16 = Formel14 & Formel15
        Worksheets(OptionType).Cells(3 + IterRauf - 1, 2 + IterRechts).Value = Formel16
    Next IterRechts
Next IterRunter
Worksheets(OptionType).Cells(3, 1).Value = 0
End Sub

```

```

Private Sub CommandButton5_Click()
'UserForm1>Underlying
    UserForm1.Hide
    myInput = Application.InputBox("1-Stock, 2-Forex, 3-Futures", , "1", , , , 2)
    If myInput = "1" Then
        Underlying = "Stock"
    ElseIf myInput = "2" Then
        Underlying = "Forex"
    ElseIf myInput = "3" Then
        Underlying = "Futures"
    End If
End Sub

```

```

End If
Call GenerateEquationsUnderlying(Underlying)
End Sub

Sub GenerateEquationsUnderlying(Underlying)
K = UserForm1.TextBox1.Value
T = UserForm1.TextBox2.Value
S0 = UserForm1.TextBox3.Value
Std = UserForm1.TextBox4.Value
r = UserForm1.TextBox5.Value
Dt = UserForm1.TextBox6.Value
Up = UserForm1.TextBox7.Value
Down = UserForm1.TextBox8.Value
Prob = UserForm1.TextBox9.Value
EinsMinusProb = UserForm1.TextBox10.Value
n = UserForm1.TextBox11.Value
IterMax = Worksheets("In").Range(n).Value
'Clear all
With Worksheets(Underlying)
.Range(.Cells(3, 1), .Cells(257, 254)).ClearContents
End With
'Definitionen
mal = "*"
Gleich = "="
Hoch = "^"
minus = "-"
plus = "+"
'Programm für Stock
For IterUnten = 0 To IterMax
Worksheets(Underlying).Cells(3 + IterUnten, 1).Value = IterUnten
For IterRechts = 0 To IterUnten
'Formel
UpStr = CStr(IterUnten - IterRechts)
DownStr = CStr(IterRechts)
Formel1 = Gleich & S0 & mal
Formel2 = Up & Hoch & UpStr & mal
Formel3 = Down & Hoch & DownStr
Worksheets(Underlying).Cells(3 + IterUnten, 2 + IterRechts).Value = Formel1 & Formel2 &
Formel3
Next IterRechts
Next IterUnten
End Sub

Private Sub UserForm_Click()

End Sub

```

## **File: OptionPricing –BlackScholes-DynamicHedging**

```

Private Sub CommandButton1_Click()
    Dim MyLoadAreaR As Range, MyLoadArea As String
    Set MyLoadAreaR = Application.InputBox(Prompt:="Select MyLoadArea", Default:="=H4:H11",
Type:=8)
    MyLoadArea = MyLoadAreaR.Address(RowAbsolute:=False, ColumnAbsolute:=False)
    Call MySolver(MyLoadArea)
End Sub

```

```

Function MySolver(MyLoadArea As String)
    'Solver starten
    Worksheets("B-S Mispriced Option").Activate
    SolverReset
    SolverLoad loadArea:=Range(MyLoadArea)
    SolverSolve (True)
    'balloon
    Set Myballoon = Assistant.NewBalloon
    With Myballoon
        .Heading = "Option Type"
        .Text = "Select your option type"
        .CheckBoxes(1).Text = "Put Option"
        .CheckBoxes(2).Text = "Call Option"
        .Button = msoButtonSetOK
        .Show
    If .CheckBoxes(1).Checked Then
        Myanswer = 1
    End If
    If .CheckBoxes(2).Checked Then
        Myanswer = 2
    End If
    End With
    'load area setzieren
    MySplit = Strings.Split(MyLoadArea, ":")
    MyLine = Strings.StrReverse(MySplit(0))
    MyLine = Conversion.Val(MyLine)
    MyLine = Strings.StrReverse(MyLine)
    MySaveArea = "F" & MyLine
    'Ergebnisse links nach rechts übertragen
    If Myanswer = 1 Then
        Worksheets("B-S Mispriced Option").Range(MySaveArea).Activate
        ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(19, 2).Value
    ElseIf Myanswer = 2 Then
        Worksheets("B-S Mispriced Option").Range(MySaveArea).Activate
        ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(18, 2).Value
    End If
    ActiveCell.Offset(2, 0).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(2, 2).Value
    ActiveCell.Offset(1, 0).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(3, 2).Value
    ActiveCell.Offset(1, 0).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(4, 2).Value
    ActiveCell.Offset(1, 0).Activate
    ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(10, 2).Value
    ActiveCell.Offset(1, 0).Activate

```

```

        ActiveCell.Value = Worksheets("B-S Mispriced Option").Cells(6, 2).Value
End Function

Private Sub CommandButton2_Click()
    Dim MyLoadAreaR As Range
    Dim MyLoadArea, MySRange, MyPRange As String
    Dim MySValue, MyPValue As Double
    'input
    Set MyLoadAreaR = Application.InputBox(Prompt:="Select MyLoadArea", Default:="=E42:E45",
Left:=80, Top:=250, Type:=8)
    MyLoadArea = MyLoadAreaR.Address(RowAbsolute:=False, ColumnAbsolute:=False)
    'load area setzieren
    'Bem.: Andere Möglichkeit mit Worksheets.Range.Activate und ActiveCell.Offset.Activate wie oben
    MyFirstFieldA1 = Strings.Split(MyLoadArea, ":")
    MyFirstFieldR1C1 = Application.ConvertFormula(MyFirstFieldA1, xlA1, xlR1C1, xlAbsolute)
    MyFirstFieldSplitR1C1 = Strings.Split(MyFirstFieldR1C1(1), "C")
    MyFirstFieldSplitR1 = Strings.Split(MyFirstFieldSplitR1C1(0), "R")
    MyFirstFieldR = MyFirstFieldSplitR1(1)
    MyFirstFieldC = MyFirstFieldSplitR1C1(1)
    MyTRange = Application.ConvertFormula("R" & CStr(MyFirstFieldR + 0) & "C" &
CStr(MyFirstFieldC), xlR1C1, xlA1)
    MySRange = Application.ConvertFormula("R" & CStr(MyFirstFieldR + 1) & "C" &
CStr(MyFirstFieldC), xlR1C1, xlA1)
    MyPRange = Application.ConvertFormula("R" & CStr(MyFirstFieldR + 2) & "C" &
CStr(MyFirstFieldC), xlR1C1, xlA1)
    MyHRange = Application.ConvertFormula("R" & CStr(MyFirstFieldR + 3) & "C" &
CStr(MyFirstFieldC), xlR1C1, xlA1)
    'Std Wert holen
    MySValue = Worksheets("B-S Mispriced Option").Range(MyTRange).Value
    'Std Wert setzen
    Worksheets("B-S Mispriced Option").Range("B6").Value = MySValue
    'S Wert holen
    MySValue = Worksheets("B-S Mispriced Option").Range(MySRange).Value
    'S Wert setzen
    Worksheets("B-S Mispriced Option").Range("B5").Value = MySValue
    'P Wert holen
    MyPValue = Worksheets("B-S Mispriced Option").Range("B19").Value
    'P Wert setzen
    Worksheets("B-S Mispriced Option").Range(MyPRange).Value = MyPValue
    'H Wert holen
    MyHValue = Worksheets("B-S Mispriced Option").Range("B24").Value
    'QP Wert holen
    MyQPValue = Worksheets("B-S Mispriced Option").Range("E40").Value
    'QS Wert berechnen
    MyQSValue = WorksheetFunction.Round((MyQPValue * (-1) * MyHValue), 0)
    'QS Wert setzen
    Worksheets("B-S Mispriced Option").Range(MyHRange).Value = MyQSValue
End Sub

```

## **File: Anna\_Task2**

```

Sub ActionList_Part1()

```

```

'msgBox to activate the right sheet, &vbCrLf &
Response = MsgBox( _
"Please select the Excel sheet you want the macro to be applied to!" & Chr(13) & _
" " & Chr(13) & _
"Click Yes: if the right sheet is already selected" & Chr(13) & _
"      Otherwise" & Chr(13) & _
"Click No: and select the Excel sheet by clicking on its tab at the" & Chr(13) & _
"      bottom and then restart the macro", _
vbYesNo)

If Response = vbNo Then
    Exit Sub
End If

'rename the sheet
Dim Sh As String
ActiveSheet.Name = "Action list"
Sh = ActiveSheet.Name

'rename the table object
Dim tbl As String
Cells(1, 1).Select
ActiveSheet.ListObjects(1).Name = "Action_list_table"
tbl = ActiveSheet.ListObjects(1).Name

'change the table style
Cells.Select
Sheets(Sh).ListObjects(tbl).TableStyle = ""
With Selection
    .HorizontalAlignment = xlGeneral
    .VerticalAlignment = xlTop
    .WrapText = True
    .Orientation = 0
    .AddIndent = False
    .IndentLevel = 0
    .ShrinkToFit = False
    .ReadingOrder = xlContext
    .MergeCells = False
End With

'delete columns J and K
Columns("J:K").Select
On Error Resume Next
Range(tbl & "[[#Headers],[Path]]").Activate
On Error GoTo 0
Selection.Delete Shift:=xlToLeft
Range(tbl & "[[#Headers],[DeCo Date]]").Select

'remove autofilter if any
Cells.Select
On Error Resume Next
Selection.AutoFilter

```

On Error GoTo 0

'select the table and clear the sort fields in the sort dialog box

Range(tbl & "[#All]").Select

ActiveWorkbook.Worksheets("Action list").ListObjects(tbl).Sort. \_  
SortFields.Clear

'add a level in the sort dialog box: by Deco Date values; oldest newest

ActiveWorkbook.Worksheets("Action list").ListObjects(tbl).Sort. \_  
SortFields.Add Key:=Range(tbl & "[DeCo Date]"), SortOn:= \_  
xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal

'add a level in the sort dialog box: by Deco Topic values; A to Z

ActiveWorkbook.Worksheets("Action list").ListObjects(tbl).Sort. \_  
SortFields.Add Key:=Range(tbl & "[DeCo Topic]"), SortOn:= \_  
xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal

'add a level in the sort dialog box: by Due Date values; oldest newest

ActiveWorkbook.Worksheets("Action list").ListObjects(tbl).Sort. \_  
SortFields.Add Key:=Range(tbl & "[Due Date]"), SortOn:= \_  
xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal

'add a level in the sort dialog box: Assigned To values; A to Z

ActiveWorkbook.Worksheets("Action list").ListObjects(tbl).Sort. \_  
SortFields.Add Key:=Range(tbl & "[Assigned To]"), SortOn:= \_  
xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal

'Apply the sort dialog box

With ActiveWorkbook.Worksheets("Action list").ListObjects(tbl). \_  
Sort  
.Header = xlYes  
.MatchCase = False  
.Orientation = xlTopToBottom  
.SortMethod = xlPinYin  
.Apply  
End With

'msg box to enter previous deco date

MsgBox ("Please click ok," & Chr(13) & \_  
"set the filter for the column Due Date manually" & Chr(13) & \_  
"and start the Macro ActionList\_Part2")

End Sub

Sub ActionList\_Part2()

'msgBox to activate the right sheet, &vbCrLf &

Response = MsgBox( \_  
"Please select the Excel sheet you want the macro to be applied to!" & Chr(13) & \_  
" " & Chr(13) & \_  
"Click Yes: if the right sheet is already selected" & Chr(13) & \_  
" Otherwise" & Chr(13) & \_  
"Click No: and select the Excel sheet by clicking on its tab at the" & Chr(13) & \_  
" bottom and then restart the macro", \_

```

vbYesNo)

If Response = vbNo Then
    Exit Sub
End If

'rename the sheet
Dim Sh As String
ActiveSheet.Name = "Action list"
Sh = ActiveSheet.Name

'rename the table object
Dim tbl As String
Cells(1, 1).Select
ActiveSheet.ListObjects(1).Name = "Action_list_table"
tbl = ActiveSheet.ListObjects(1).Name

'set autofilter column I
'this is done manually between the macros ActionList_Part1 and ActionList_Part2

'set autofilter column E
ActiveSheet.ListObjects(tbl).Range.AutoFilter Field:=5, _
    Criteria1:="<>"

'select column E and F and change format of the filtered cells
Range("E2:F10000").Select
With Selection.Font
    .Color = -65536
    .TintAndShade = 0
End With
Selection.Font.Bold = True

'remove Autofilter column E
ActiveSheet.ListObjects(tbl).Range.AutoFilter Field:=5

'set autofilter column F
ActiveSheet.ListObjects(tbl).Range.AutoFilter Field:=6, _
    Criteria1:="Completed"

'select column E and F and change format of the filtered cells
Range("E2:F10000").Select
With Selection.Font
    .Color = -65536
    .TintAndShade = 0
End With
Selection.Font.Bold = True

'Remove autofilter column F
ActiveSheet.ListObjects(tbl).Range.AutoFilter Field:=6

'remove autofilter column I
ActiveSheet.ListObjects(tbl).Range.AutoFilter Field:=9

```



```

'remove column H and I
Columns("H:I").Select
Range(tbl & "[[#Headers],[Modified]]").Activate
Selection.Delete Shift:=xlToLeft

'correct column D
Dim Row As Long
Dim SplitArr As Variant
Dim sSplitArr As String
Dim iter As Integer
Range(tbl & "[[#Headers],[Assigned To]]").Select
Row = 1
Do Until ActiveCell.Offset(Row, 0) = Empty
    SplitArr = Split(ActiveCell.Offset(Row, 0), "#")
    iter = 0
    sSplitArr = Empty
    For iter = LBound(SplitArr) To UBound(SplitArr) Step 2
        sSplitArr = sSplitArr & " " & SplitArr(iter)
    Next iter
    ActiveCell.Offset(Row, 0) = sSplitArr
    Row = Row + 1
Loop

'Select all and change the format
Cells.Select
With Selection.Font
    .Name = "Arial"
    .Size = 10
    .Strikethrough = False
    .Superscript = False
    .Subscript = False
    .OutlineFont = False
    .Shadow = False
    .Underline = xlUnderlineStyleNone
    .TintAndShade = 0
    .ThemeFont = xlThemeFontNone
End With

'select the whole table and add all borders
Range(tbl & "[#All]").Select
'diagonal none
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
'outside left
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
'outside top
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous

```

```

        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    'outside bottom
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    'outside right
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    'inside vertical
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    'inside horizontal
    With Selection.Borders(xlInsideHorizontal)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With

    'change the column width
    Columns("A:A").ColumnWidth = 11.5
    Columns("B:B").ColumnWidth = 23
    Columns("C:C").ColumnWidth = 24
    Columns("D:D").ColumnWidth = 15.5
    Columns("E:E").ColumnWidth = 32
    Columns("F:F").ColumnWidth = 10.5
    Columns("G:G").ColumnWidth = 10.5

    'wrap the text of all cells
    Cells.Select
    Selection.WrapText = False
    Selection.WrapText = True
    'workaround because WrapText is ignored when pasting the table into word
    Range(tbl & "[#All]").Select
    For Each Cell In Selection.Cells
        If Cell = Empty Then
            Cell.Select
            Selection.Value = Space(1) & ""
        End If
    Next Cell

```

Next

'select the whole table and copy the table into a word document

Range(tbl & "[#All]").Select

Selection.Copy

Application.CutCopyMode = False

'add a word application object

Dim wrdApp As Word.Application

Set wrdApp = CreateObject("Word.Application")

wrdApp.Visible = True

'add an empty word document

Dim wrdDoc As Word.Document

Set wrdDoc = wrdApp.Documents.Add(Template:="Normal", NewTemplate:=False,  
DocumentType:=0)

Application.Wait (Now + TimeValue("0:00:01"))

'format

wrdDoc.Activate

With ActiveDocument.PageSetup

.LineNumbering.Active = False

.Orientation = wdOrientLandscape

.TopMargin = CentimetersToPoints(1.5)

.BottomMargin = CentimetersToPoints(1.75)

.LeftMargin = CentimetersToPoints(1.5)

.RightMargin = CentimetersToPoints(1)

.Gutter = CentimetersToPoints(0)

.HeaderDistance = CentimetersToPoints(1.25)

.FooterDistance = CentimetersToPoints(1.25)

.PageWidth = CentimetersToPoints(29.7)

.PageHeight = CentimetersToPoints(21)

.FirstPageTray = wdPrinterDefaultBin

.OtherPagesTray = wdPrinterDefaultBin

.SectionStart = wdSectionNewPage

.OddAndEvenPagesHeaderFooter = False

.DifferentFirstPageHeaderFooter = False

.VerticalAlignment = wdAlignVerticalTop

.SuppressEndnotes = False

.MirrorMargins = False

.TwoPagesOnOne = False

.BookFoldPrinting = False

.BookFoldRevPrinting = False

.BookFoldPrintingSheets = 1

.GutterPos = wdGutterPosLeft

End With

Application.Wait (Now + TimeValue("0:00:01"))

'in excel select the whole table and copy the table into a word document

ActiveWorkbook.Activate

Sheets("Action list").Activate

Range(tbl & "[#All]").Select

Selection.Copy

```

'paste the table into a word document
wrdApp.Activate
wrdDoc.Select
wrdApp.Selection.PasteExcelTable False, False, True

'do some formatting on the table in word
Dim tbl1 As Word.Table
Set tbl1 = wrdApp.Selection.Tables(1)
With tbl1
'.Style = "Table Professional"
.ApplyStyleHeadingRows = True
.ApplyStyleLastRow = True
.ApplyStyleFirstColumn = True
.ApplyStyleLastColumn = True
.Rows(1).Shading.Texture = wdTextureNone
.Rows(1).Shading.ForegroundPatternColor = wdColorAutomatic
.Rows(1).Shading.BackgroundPatternColor = -603930625
.Rows(1).Range.Font.Bold = True
.Rows(1).Range.Font.Size = 10
.Rows(1).HeadingFormat = True
.Rows(1).HeightRule = wdRowHeightExactly
.Rows(1).Height = 25
End With

Set wrdDoc = Nothing
Set wrdApp = Nothing
Set tbl1 = Nothing

End Sub

```

## **File: API VBA GenerateSmartchoiceReports**

```

Sub EmailOrder()
'Versendet das Aktuelle Workbook als Anhang mit Windows Mail
'Erzeugt eine Warnung. Diese lässt sich ein- und ausschalten über:
Extras>Optionen>Sicherheit>Virenschutz>Warnen...

```

```

With ActiveWorkbook
.SendMail Recipients:="roland-benz@hispeed.ch", _
subject:="Testing"
End With

```

```

End Sub

```

'Dieses Beispiel sendet eine e-Mail über Outlook:  
 'Zum Anzeigen dieser Einstellungen führen Sie folgende Aktionen aus:

'Klicken Sie im Menü Extras auf Vertrauensstellungscenter.  
 'Klicken Sie auf Programmgesteuerter Zugriff.

```

Sub Mail_senden()
    Dim olApp As Object
    Set olApp = CreateObject("Outlook.Application")
    With olApp.CreateItem(0)
        'Empfänger
        .Recipients.Add "roland-benz@hispeed.ch"
        'Betreff
        .subject = "Test-Mail"
        'Nachricht
        .body = "Das ist eine e-Mail" & Chr(13) & _
            "Viele Grüße..." & Chr(13) & Chr(13)
        'Lesebestätigung aus
        .ReadReceiptRequested = False
        'Dateianhang
        .Attachments.Add "C:\Users\Benzro\Desktop\A.txt"
        .send
    End With
    Set olApp = Nothing
End Sub

```

'You can copy the script and attach to any of your report  
 'but make sure that inside the VBA Editor select Tools->References and  
 'select Microsoft CDO1.21 Library before copying the script.

```

Sub SendMail()

    Dim objSession As MAPI.Session ' Local
    Dim objMessage As Message ' local
    Dim objRecip As Recipient
    On Error GoTo error_olemsg
    Dim doc As busobj.IDocument
    Dim rep As busobj.Report
    Dim DPName As String
    Dim test As Boolean

    Set objSession = CreateObject("MAPI.Session")
    objSession.Logon profileName:="bo_admin", NewSession:=False, showDialog:=False

    If objSession Is Nothing Then
        Err.Raise 10, "MA MACRO", "must first log on; use Session->Logon"
        Exit Sub
    End If

    Set objMessage = objSession.Outbox.Messages.Add
    If objMessage Is Nothing Then
        Err.Raise 11, "MA MACRO", "could not create a new message in the Outbox"
        Exit Sub
    End If

    With objMessage ' message object
        ' Substitutue this with your subject
    End With

```

```

.subject = "Resort -Monthly Report"
' Substitue with your the message in body part of the mail
.Text = "The Monthly reports for " & Format(Now, "mmm") & " is attached herewith."
For i = 1 To ThisDocument.DataProviders.Count
If ActiveDocument Is Nothing Then
MsgBox "NO Active Document to refresh"
Else
Set doc = ActiveDocument
If Not doc.IsAddin Then

'use this for converting to csv

DPName = "C:\\" + DataProviders.Item(i).Name
test = DataProviders.Item(i).ConvertTo(boExpAsciiCSV, 1, DPName)

'use this for converting to pdf format

Else
End If
End If
Set objAttach = .Attachments.Add ' add the attachment

If objAttach Is Nothing Then
Err.Raise 12, "MA MACRO", "Unable to create new Attachmentobject"
Exit Sub
End If

With objAttach

.Name = DataProviders.Item(i).Name & ".csv"
.Source = "C:\\" & DataProviders.Item(i).Name & ".csv"

End With

.Update ' update message to save attachment in MAPI system

Next i

Set objRecip = .Recipients.Add
With objRecip
objRecip.Name = ("MAILID") 'substitue with the mailid of the recipient or groupname
objRecip.Type = CdoTo
objRecip.Resolve
End With

' use this for sending to a recipient as cc

'Set objRecip = .Recipients.Add
'With objRecip

```

```

' objRecip.Name = ("ddas")
' objRecip.Type = CdoCc
' objRecip.Resolve
' End With

.Update
' update message to save attachment in MAPI system
.send showDialog:=False
End With
For i = 1 To ThisDocument.DataProviders.Count
Kill "C:\\" & DataProviders.Item(i).Name & ".csv"
Next i
objSession.Logoff
Exit Sub

error_olemsg:
'MsgBox "Error " & Str(Err) & ": " & Error$(Err)
Err.Raise 13, "MA MACRO", "Error " & Str(Err) & ": " & Error$(Err)
Resume Next

End Sub

'Before you start, add a reference to the MAPI controls library.
'This is probably somewhere such as C:\Windows\System\MSMAPI32.OCX.
' Send an email message.
Public Sub SendEmail(ByVal to_name As String, ByVal _
to_address As String, ByVal cc_name As String, ByVal _
cc_address As String, ByVal subject As String, ByVal _
body As String)
Dim mapi_session As MSMAPI.MAPISession
Dim mapi_messages As MSMAPI.MAPIMessages

'Debug.Print "To: " & to_name & "<" & to_address & ">"
'Debug.Print "Cc: " & cc_name & "<" & cc_address & ">"
'Debug.Print "Subject: " & subject
'Debug.Print "Body: " & body
'Debug.Print
' "=====

On Error GoTo MailError

Set mapi_session = New MSMAPI.MAPISession
With mapi_session
.LogonUI = False
' Fill in username and password
' if necessary on this mail server.
'.username = "username"
'.password = "password"
.SignOn
End With

```

```

Set mapi_messages = New MSMAPI.MAPIMessages
With mapi_messages
    .SessionID = mapi_session.SessionID
    .Compose

    .RecipIndex = 0
    .RecipDisplayName = to_name
    .RecipAddress = to_address
    .RecipType = mapToList

    .RecipIndex = 1
    .RecipDisplayName = cc_name
    .RecipAddress = cc_address
    .RecipType = mapCcList

    .AddressResolveUI = False
    .MsgSubject = subject
    .MsgNoteText = body
    .send False
End With

mapi_session.SignOff
Exit Sub

MailError:
    MsgBox Err.Description
    Exit Sub
End Sub

Sub S1()
    SendEmail "Roland Benz", "roland-benz@hispeed.ch", "", "", "Testing", "Dies ist ein Test"
End Sub

Declare Function MoveWindow Lib "user32.dll" ( _
    ByVal hwnd As Long, _
    ByVal x As Long, _
    ByVal y As Long, _
    ByVal nWidth As Long, _
    ByVal nHeight As Long, _
    ByVal bRepaint As Long) As Long

Private Declare Sub Sleep Lib "kernel32" ( _
    ByVal dwMilliseconds As Long)

Private Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" ( _
    ByVal lpClassName As String, _
    ByVal lpWindowName As String) As Long

```



\*\*\*\*Tools>References: Microsoft Internet Controls

```
'returns new instance of Internet Explorer
Function GetNewIE() As SHDocVw.InternetExplorer
    'create new IE instance
    Set GetNewIE = New SHDocVw.InternetExplorer
    'start with a blank page
    GetNewIE.Navigate2 "about:Blank"
End Function
```

```
'loads a web page and returns True or False depending on
'whether the page could be loaded or not
Function LoadWebPage(i_IE As SHDocVw.InternetExplorer, _
    i_URL As String) As Boolean
    With i_IE
        'open page
        .Navigate i_URL
        'wait until IE finished loading the page
        Do While .ReadyState <> READYSTATE_COMPLETE
            Application.Wait Now + TimeValue("0:00:01")
        Loop
        'check if page could be loaded
        If .Document.URL = i_URL Then
            LoadWebPage = True
        End If
    End With
End Function
```

```
'finds an open IE site by checking the URL
Function GetOpenIEByURL(ByVal i_URL As String) _
    As SHDocVw.InternetExplorer
```

```
Dim objShellWindows As New SHDocVw.ShellWindows
```

```
'ignore errors when accessing the document property
On Error Resume Next
'loop over all Shell-Windows
For Each GetOpenIEByURL In objShellWindows
    'if the document is of type HTMLDocument, it is an IE window
    If TypeName(GetOpenIEByURL.Document) = "HTMLDocument" Then
        'check the URL
        If GetOpenIEByURL.Document.URL = i_URL Then
            'leave, we found the right window
            Exit Function
        End If
    End If
Next
End Function
```

```
'finds an open IE site by checking the title
Function GetOpenIEByTitle(i_Title As String, _
```

```
Optional ByVal i_ExactMatch As Boolean = True) _  
As SHDocVw.InternetExplorer
```

```
Dim objShellWindows As New SHDocVw.ShellWindows
```

```
If i_ExactMatch = False Then i_Title = "*" & i_Title & "*"  
'ignore errors when accessing the document property  
On Error Resume Next  
'loop over all Shell-Windows  
For Each GetOpenIEByTitle In objShellWindows  
'if the document is of type HTMLDocument, it is an IE window  
If TypeName(GetOpenIEByTitle.Document) = "HTMLDocument" Then  
'check the title  
If GetOpenIEByTitle.Document.Title Like i_Title Then  
'leave, we found the right window  
Exit Function  
End If  
End If  
Next  
End Function
```

```
Function OpenIE_CheckIfPageNameOpen_OpenURL(myPageTitle As String, myPageURL As String) _  
As SHDocVw.InternetExplorer
```

```
'IE object instantiation  
Dim myIE As SHDocVw.InternetExplorer
```

```
'check if page is already open  
Set myIE = GetOpenIEByTitle(myPageTitle, False)
```

```
'check if page is already open  
Set myIE = GetOpenIEByURL(myPageURL)
```

```
'if the page is not open then try to open it  
If myIE Is Nothing Then  
'page isn't open yet  
'create new IE instance  
Set myIE = GetNewIE  
'make IE window visible  
myIE.Visible = True  
'load page  
If LoadWebPage(myIE, myPageURL) = False Then  
'page wasn't loaded  
MsgBox "Couldn't open page"  
Exit Function  
End If  
End If  
Set OpenIE_CheckIfPageNameOpen_OpenURL = myIE  
End Function
```

```
Sub ExplorerTest()  
'SmC IE page title (used to check if already open)
```

```
Const myPageTitle As String = "SmartChoice"
```

```
'SmC stage (used to open the site)
```

```
Const myPageURL As String =
```

```
"http://opxscs.eame.syngenta.org/STAGE/OPX2/frgolappschs02.eame.syngenta.org:8404/HOME?  
dispatched=http://opxscs.eame.syngenta.org/STAGE/OPX2/15.141.4.48:8100/"
```

```
'SmC prod (used to open the site)
```

```
'Const myPageURL As String =
```

```
"http://opxscp.eame.syngenta.org/PROD/OPX2/frgolappschp01.eame.syngenta.org:8401/HOME?  
dispatched=http://opxscp.eame.syngenta.org/PROD/OPX2/15.141.4.50:8100/"
```

```
'IE object instantiation
```

```
Dim myIE As SHDocVw.InternetExplorer
```

```
'open SmC
```

```
'Call OpenIE_CheckIfPageNameOpen_OpenURL(myPageTitle, myPageURL)
```

```
Set myIE = OpenIE_CheckIfPageNameOpen_OpenURL(myPageTitle, myPageURL)
```

```
'move and resize the window
```

```
Dim nhWnd As Long
```

```
nhWnd1 = FindWindow(vbNullString, "SmartChoice - Windows Internet Explorer provided by  
Syngenta")
```

```
If nhWnd1 <> 0 Then
```

```
MoveWindow nhWnd1, 0, 0, 1200, 900, 1
```

```
End If
```

```
End Sub
```

```
Option Explicit
```

```
Private Declare Sub Sleep Lib "kernel32.dll" ( _  
ByVal dwMilliseconds As Long)
```

```
Private Declare Function GetCursorPos Lib "user32.dll" ( _  
ByRef lpPoint As POINTAPI) As Long
```

```
Private Declare Function SetCursorPos Lib "user32.dll" ( _  
ByVal x As Long, _  
ByVal y As Long) As Long
```

```
Private Declare Function GetAsyncKeyState Lib "user32.dll" (ByVal vKey As Long) As Long
```

```
Private Type POINTAPI
```

```
x As Long
```

```
y As Long
```

```
End Type
```

```
Const VK_LBUTTON = &H1 ' Linker Mausbutton
```

```
Const VK_RBUTTON = &H2 ' Rechter Mausbutton
```

```
Const KEYEVENTF_KEYUP = &H2 ' Die angegebene Taste wird losgelassen
```

```

Public Sub maus_spazieren_Fahren()
Dim myPos As POINTAPI
Dim Ziel_X As Long
Dim Ziel_Y As Long
Dim L As Long
Dim Click
Ziel_X = 10
Ziel_Y = 100
GetCursorPos myPos
If myPos.x >= Ziel_X Then
    For L = myPos.x To Ziel_X Step -1
        Sleep 5
        SetCursorPos L, myPos.y
    Next
Else:
    For L = Ziel_X To myPos.x
        Sleep 5
        SetCursorPos L, myPos.y
    Next
End If
If myPos.y >= Ziel_Y Then
    For L = myPos.y To Ziel_Y Step -1
        Sleep 5
        SetCursorPos Ziel_X, L
    Next
Else:
    For L = myPos.y To Ziel_Y
        Sleep 5
        SetCursorPos Ziel_X, L
    Next
End If
Click = GetAsyncKeyState(&H1)
MsgBox Click
End Sub

```

Option Explicit

```

Private Declare Function SetCursorPos Lib "user32.dll" ( _
    ByVal x As Long, _
    ByVal y As Long) As Long

Private Declare Sub Sleep Lib "kernel32.dll" ( _
    ByVal dwMilliseconds As Long)

Private Declare Function GetCursorPos Lib "user32.dll" ( _
    ByRef lpPoint As POINTAPI) As Long

Private Type POINTAPI

```

```
x As Long
y As Long
End Type
```

```
Private Declare Sub mouse_event Lib "user32" _
    (ByVal dwFlags As Long, ByVal dx As Long, _
    ByVal dy As Long, ByVal cButtons As Long, _
    ByVal dwExtraInfo As Long)
```

```
Public Const MOUSE_LEFT = 0
Public Const MOUSE_MIDDLE = 1
Public Const MOUSE_RIGHT = 2
```

'Die nachfolgende Prozedur simuliert den gewünschten Mausklick.

```
Public Sub SendMausklick(ByVal mButton As Long)
    Const MOUSEEVENTF_LEFTDOWN = &H2
    Const MOUSEEVENTF_LEFTUP = &H4
    Const MOUSEEVENTF_MIDDLEDOWN = &H20
    Const MOUSEEVENTF_MIDDLEUP = &H40
    Const MOUSEEVENTF_RIGHTDOWN = &H8
    Const MOUSEEVENTF_RIGHTUP = &H10

    If (mButton = MOUSE_LEFT) Then
        Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
        Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
    ElseIf (mButton = MOUSE_MIDDLE) Then
        Call mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
        Call mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
    Else
        Call mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
        Call mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
    End If
End Sub
```

```
Sub SendMausDoppelklick(ByVal mButton As Long)
    SendMausklick (mButton)
    SendMausklick (mButton)
End Sub
```

```
Sub MainMausPlatzieren()
    Dim myAktuellPos As POINTAPI
    Dim myZielPos As POINTAPI
    Dim Ziel_X, Ziel_Y As Long
    Dim Aktuell_X, Aktuell_Y As Long
    'aktuelle pos abfragen
    GetCursorPos myAktuellPos
    Aktuell_X = myAktuellPos.x
    Aktuell_Y = myAktuellPos.y
    MsgBox Aktuell_X & " " & Aktuell_Y
    'neue pos setzen
    Ziel_X = 10
```

```

Ziel_Y = 100
SetCursorPos Ziel_X, Ziel_Y
'aktuelle pos abfragen
GetCursorPos myAktuellPos
Aktuell_X = myAktuellPos.x
Aktuell_Y = myAktuellPos.y
MsgBox Aktuell_X & " " & Aktuell_Y
End Sub

Sub MainSendMousclicks()
SendMausklick (MOUSE_LEFT)
SendMausDoppelklick (MOUSE_LEFT)
End Sub

Sub ApplikationStartenUndSendKeys()
' Notepad starten und Dialog "Seite einrichten" aufrufen
Dim ApplID As Long

ApplID = Shell("C:\Windows\System32\notepad.exe", vbNormalFocus)
DoEvents

' NotePad aktivieren
AppActivate ApplID

' Alt+d (Menü DATEI)
SendKeys "%d", True

' r (Seiten einrichten)
SendKeys "r", True

' Alt+k (Kopfzeile)
SendKeys "%k", True

' Text schreiben
SendKeys "Test-Kopfzeile", True

' Dialog beenden (OK-Schaltfläche per Alt+O auslösen)
SendKeys "{ENTER}"

' oder anstelle OK, Dialog per Alt+F4 schließen
'SendKeys "%{F4}"

End Sub

Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" ( _
    ByVal hwnd As Long, _
    ByVal lpOperation As String, _
    ByVal lpFile As String, _
    ByVal lpParameters As String, _
    ByVal lpDirectory As String, _
    ByVal nShowCmd As Long) As Long

```

```
Private Const SW_HIDE = 0
Private Const SW_MAXIMIZE = 3
Private Const SW_MINIMIZE = 6
Private Const SW_NORMAL = 1
Private Const SW_SHOW = 5
Private Const SW_RESTORE = 9
Private Const SW_SHOWMAXIMIZED = 3
Private Const SW_SHOWMINIMIZED = 2
Private Const SW_SHOWMINNOACTIVE = 7
Private Const SW_SHOWNA = 8
Private Const SW_SHOWNOACTIVATE = 4
Private Const SW_SHOWNORMAL = 1
```

```
Private Const ERROR_BAD_FORMAT = 11&
Private Const SE_ERR_ACCESSDENIED = 5
Private Const SE_ERR_ASSOCINCOMPLETE = 27
Private Const SE_ERR_DDEBUSY = 30
Private Const SE_ERR_DDEFAIL = 29
Private Const SE_ERR_DDETIMEOUT = 28
Private Const SE_ERR_DLLNOTFOUND = 32
Private Const SE_ERR_FNF = 2
Private Const SE_ERR_NOASSOC = 31
Private Const SE_ERR_OOM = 8
Private Const SE_ERR_PNF = 3
Private Const SE_ERR_SHARE = 26
```

```
Const HH_DISPLAY_TOPIC = &H0
Const HH_HELP_CONTEXT = &HF
```

```
Declare Function MoveWindow Lib "user32.dll" ( _
    ByVal hwnd As Long, _
    ByVal x As Long, _
    ByVal y As Long, _
    ByVal nWidth As Long, _
    ByVal nHeight As Long, _
    ByVal bRepaint As Long) As Long
```

```
Private Declare Sub Sleep Lib "kernel32" ( _
    ByVal dwMilliseconds As Long)
```

```
Private Declare Function FindWindow Lib "user32" _
Alias "FindWindowA" ( _
    ByVal lpClassName As String, _
    ByVal lpWindowName As String) As Long
```

```
Sub MyMoveShellExecuteNotepad()
    Dim retVal, rval, x, y, z As Long
    Dim AppID As Long
    retVal = ShellExecute(Application.hwnd, "open", _
        "C:\Users\Benzro\Desktop\A.txt", "", _
        "", SW_NORMAL)
```

DoEvents

Sleep 100 'Application.Wait (Now + TimeValue("0:00:1"))

nhWnd = FindWindow(vbNullString, "A.txt - Editor")

If nhWnd <> 0 Then

    rval = MoveWindow(nhWnd, 150, 150, 500, 500, 1)

End If

Sleep 100

End Sub

Private Sub MyMoveShellNotepad()

Dim nhWnd As Long

nhWnd = Shell("C:\Windows\System32\notepad.exe", vbNormalFocus)

'nhWnd = FindWindow(vbNullString, "Unbenannt - Editor")'German version

nhWnd = FindWindow(vbNullString, "Untitled - Notepad") 'English version

If nhWnd <> 0 Then

    MoveWindow nhWnd, 150, 150, 500, 500, 1

End If

End Sub

Private Sub MyMoveShellExplorer()

Dim nhWnd As Long

'nhWnd = Shell("C:\Program Files (x86)\Internet Explorer\iexplore.exe", vbNormalFocus)

'nhWnd1 = FindWindow(vbNullString, "Google - Windows Internet Explorer provided by Syngenta")

nhWnd1 = FindWindow(vbNullString, "SmartChoice - Windows Internet Explorer provided by Syngenta")

If nhWnd <> 0 Then

    MoveWindow nhWnd1, 0, 0, 1200, 900, 1

End If

End Sub

Public Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" \_

(ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, \_

ByVal lpParameters As String, ByVal lpDirectory As String, \_

ByVal nShowCmd As Long) As Long

Public Const SW\_HIDE = 0           ' Versteckt öffnen

Public Const SW\_MAXIMIZE = 3       ' Maximiert öffnen

Public Const SW\_MINIMIZE = 6       ' Minimiert öffnen

Public Const SW\_NORMAL = 1

Public Const SW\_RESTORE = 9

Public Const SW\_SHOWMAXIMIZED = 3

Public Const SW\_SHOWMINIMIZED = 2

Public Const SW\_SHOWMINNOACTIVE = 7

Public Const SW\_SHOWNOACTIVATE = 4

Sub DateiOeffnen()



```

' Text-Datei öffnen:
Call ShellExecute(Application.hwnd, "open", _
    "C:\Users\Benzro\Desktop\A.txt", "", _
    "", SW_NORMAL)
' Word-Datei öffnen:
'Call ShellExecute(Me.hWnd, "open", _
    "C:\MeinPfad\Mein.Doc", "", _
    "", SW_NORMAL)

' Excel-Datei im Hintergrund drucken:
'Call ShellExecute(Me.hWnd, "print", _
    "C:\MeinPfad\Mein.XLS", _
    "", "", SW_HIDE)

' Explorer-Fenster mit einem vorgegebenen Pfad öffnen:
'Call ShellExecute(Me.hWnd, "explore", _
    "", "C:\MeinPfad\", _
    "", SW_NORMAL)

' Anwendung in einem bestimmten Verzeichnis ausführen, Fenster maximieren:
'Call ShellExecute(Me.hWnd, "Print", _
    "C:\MeinPfad\Mein.XLS", "C:\MeinAndererPfad", _
    "", SW_MAXIMIZE)

```

End Sub

```

'---window
Private Const SW_HIDE = 0
Private Const SW_MAXIMIZE = 3
Private Const SW_MINIMIZE = 6
Private Const SW_NORMAL = 1
Private Const SW_SHOW = 5
Private Const SW_RESTORE = 9
Private Const SW_SHOWMAXIMIZED = 3
Private Const SW_SHOWMINIMIZED = 2
Private Const SW_SHOWMINNOACTIVE = 7
Private Const SW_SHOWNA = 8
Private Const SW_SHOWNOACTIVATE = 4
Private Const SW_SHOWNORMAL = 1

Private Const ERROR_BAD_FORMAT = 11&
Private Const SE_ERR_ACCESSDENIED = 5
Private Const SE_ERR_ASSOCINCOMPLETE = 27
Private Const SE_ERR_DDEBUSY = 30
Private Const SE_ERR_DDEFAIL = 29
Private Const SE_ERR_DDETIMEOUT = 28
Private Const SE_ERR_DLLNOTFOUND = 32
Private Const SE_ERR_FNF = 2
Private Const SE_ERR_NOASSOC = 31
Private Const SE_ERR_OOM = 8
Private Const SE_ERR_PNF = 3

```

Private Const SE\_ERR\_SHARE = 26

Const HH\_DISPLAY\_TOPIC = &H0

Const HH\_HELP\_CONTEXT = &HF

Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" ( \_  
    ByVal hwnd As Long, \_  
    ByVal lpOperation As String, \_  
    ByVal lpFile As String, \_  
    ByVal lpParameters As String, \_  
    ByVal lpDirectory As String, \_  
    ByVal nShowCmd As Long) As Long

Declare Function MoveWindow Lib "user32.dll" ( \_  
    ByVal hwnd As Long, \_  
    ByVal x As Long, \_  
    ByVal y As Long, \_  
    ByVal nWidth As Long, \_  
    ByVal nHeight As Long, \_  
    ByVal bRepaint As Long) As Long

Private Declare Function FindWindow Lib "user32" \_  
    Alias "FindWindowA" ( \_  
    ByVal lpClassName As String, \_  
    ByVal lpWindowName As String) As Long

Private Declare Function ShowWindow \_  
    Lib "user32" \_  
    (ByVal hwnd As Long, \_  
    ByVal nCmdShow As Long) As Long

'----mouse

Private Declare Function SetCursorPos Lib "user32.dll" ( \_  
    ByVal x As Long, \_  
    ByVal y As Long) As Long

Private Declare Sub Sleep Lib "kernel32.dll" ( \_  
    ByVal dwMilliseconds As Long)

Private Declare Function GetCursorPos Lib "user32.dll" ( \_  
    ByRef lpPoint As POINTAPI) As Long

Private Type POINTAPI  
    x As Long  
    y As Long  
End Type

Private Declare Sub mouse\_event Lib "user32" \_  
    (ByVal dwFlags As Long, ByVal dx As Long, \_  
    ByVal dy As Long, ByVal cButtons As Long, \_  
    ByVal dwExtraInfo As Long)

```

Public Const MOUSE_LEFT = 0
Public Const MOUSE_MIDDLE = 1
Public Const MOUSE_RIGHT = 2

Private Sub MyMoveShellExplorer()
Dim nhWnd As Long
'Open Windows Explorer
'nhWnd = Shell("C:\Program Files (x86)\Internet Explorer\iexplore.exe", vbNormalFocus)
'nhWnd1 = FindWindow(vbNullString, "Google - Windows Internet Explorer provided by Syngenta")

'Open IE

'Find Window to get the right handel
nhWnd1 = FindWindow(vbNullString, "SmartChoice - Windows Internet Explorer provided by
Syngenta")

'Move and resize window
If nhWnd1 <> 0 Then
    MoveWindow nhWnd1, 0, 0, 1200, 900, 1
End If

'set focus to window
ShowWindow IHwnd, SW_SHOWNORMAL

'position the mouse
Dim myAktuellPos As POINTAPI
Dim myZielPos As POINTAPI
Dim Ziel_X, Ziel_Y As Long
Dim Aktuell_X, Aktuell_Y As Long
'aktuelle pos abfragen
GetCursorPos myAktuellPos
Aktuell_X = myAktuellPos.x
Aktuell_Y = myAktuellPos.y
MsgBox Aktuell_X & " " & Aktuell_Y
'neue pos setzen
Ziel_X = 24
Ziel_Y = 285
SetCursorPos Ziel_X, Ziel_Y
'aktuelle pos abfragen
GetCursorPos myAktuellPos
Aktuell_X = myAktuellPos.x
Aktuell_Y = myAktuellPos.y
MsgBox Aktuell_X & " " & Aktuell_Y

'mouse click
SendMausklick (MOUSE_LEFT)
'SendMausDoppelklick (MOUSE_LEFT)

End Sub

```

```

Public Sub SendMausklick(ByVal mButton As Long)
    Const MOUSEEVENTF_LEFTDOWN = &H2
    Const MOUSEEVENTF_LEFTUP = &H4
    Const MOUSEEVENTF_MIDDLEDOWN = &H20
    Const MOUSEEVENTF_MIDDLEUP = &H40
    Const MOUSEEVENTF_RIGHTDOWN = &H8
    Const MOUSEEVENTF_RIGHTUP = &H10

    If (mButton = MOUSE_LEFT) Then
        Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
        Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
    ElseIf (mButton = MOUSE_MIDDLE) Then
        Call mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
        Call mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
    Else
        Call mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
        Call mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
    End If
End Sub

```

'-----Mouse

```

Private Declare Function SetCursorPos Lib "user32.dll" ( _
    ByVal x As Long, _
    ByVal y As Long) As Long

```

```

Private Declare Sub Sleep Lib "kernel32.dll" ( _
    ByVal dwMilliseconds As Long)

```

```

Private Declare Function GetCursorPos Lib "user32.dll" ( _
    ByRef lpPoint As POINTAPI) As Long

```

```

Private Type POINTAPI
    x As Long
    y As Long
End Type

```

```

Private Declare Sub mouse_event Lib "user32" _
    (ByVal dwFlags As Long, ByVal dx As Long, _
    ByVal dy As Long, ByVal cButtons As Long, _
    ByVal dwExtraInfo As Long)

```

```

Public Const MOUSE_LEFT = 0
Public Const MOUSE_MIDDLE = 1
Public Const MOUSE_RIGHT = 2

```

'-----Window

```

Private Declare Function PostMessage Lib "user32" _
    Alias "PostMessageA" _
    (ByVal hwnd As Long, _
    ByVal wMsg As Long, _
    ByVal wParam As Long, _

```

ByVal lParam As Long) As Long

```
Declare Function MoveWindow Lib "user32.dll" ( _  
    ByVal hwnd As Long, _  
    ByVal x As Long, _  
    ByVal y As Long, _  
    ByVal nWidth As Long, _  
    ByVal nHeight As Long, _  
    ByVal bRepaint As Long) As Long
```

```
Private Declare Function FindWindow Lib "user32" _  
    Alias "FindWindowA" ( _  
    ByVal lpClassName As String, _  
    ByVal lpWindowName As String) As Long
```

```
' Module Name: ModFindWindowLike  
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)  
' Written 02/06/2005
```

```
Private Declare Function EnumWindows Lib "user32" _  
    (ByVal lpEnumFunc As Long, _  
    ByVal lParam As Long) As Long
```

```
Private Declare Function GetWindowText Lib "user32" _  
    Alias "GetWindowTextA" _  
    (ByVal hwnd As Long, _  
    ByVal lpString As String, _  
    ByVal cch As Long) As Long
```

```
' Custom structure for passing in the parameters in/out of the hook enumeration function  
' Could use global variables instead, but this is nicer.
```

```
Private Type FindWindowParameters
```

```
    strTitle As String 'INPUT  
    hwnd As Long      'OUTPUT
```

```
End Type
```

```
' Module Name: ModSetForegroundWindow  
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)  
' Written 02/06/2005
```

```
Private Declare Function SetForegroundWindow Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Private Declare Function GetWindowThreadProcessId Lib "user32" _  
    (ByVal hwnd As Long, _  
    lpdwProcessId As Long) As Long
```

```
Private Declare Function IsIconic Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Private Declare Function ShowWindow Lib "user32" _  
    (ByVal hwnd As Long, _  
    ByVal nCmdShow As Long) As Long
```

```
Private Declare Function AttachThreadInput Lib "user32" _  
    (ByVal idAttach As Long, _  
    ByVal idAttachTo As Long, _  
    ByVal fAttach As Long) As Long
```

```
Private Declare Function GetForegroundWindow Lib "user32" _  
    () As Long
```

```
Private Const SW_RESTORE = 9  
Private Const SW_SHOW = 5
```

```
' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel  
,  
' sTitel: muss nicht der exakte Fenstertitel sein  
'     hier kann bspw. auch nur der Anfang des Fenstertitel  
'     angegeben werden, z.B.: Fenstertitel*  
,  
' benötigte API-Deklarationen
```

```
Private Declare Function GetWindowTextLength Lib "user32" _  
    Alias "GetWindowTextLengthA" ( _  
    ByVal hwnd As Long) As Long
```

```
Private Declare Function GetWindow Lib "user32" ( _  
    ByVal hwnd As Long, _  
    ByVal wCmd As Long) As Long
```

```
Private Const GW_HWNDNEXT = 2
```

```
'-----Mouse
```

```
'Die nachfolgende Prozedur simuliert den gewünschten Mausklick.
```

```
Public Sub SendMouseClicked(ByVal mButton As Long)
```

```
    Const MOUSEEVENTF_LEFTDOWN = &H2  
    Const MOUSEEVENTF_LEFTUP = &H4  
    Const MOUSEEVENTF_MIDDLEDOWN = &H20  
    Const MOUSEEVENTF_MIDDLEUP = &H40  
    Const MOUSEEVENTF_RIGHTDOWN = &H8  
    Const MOUSEEVENTF_RIGHTUP = &H10
```

```
    If (mButton = MOUSE_LEFT) Then
```

```
        Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
```

```
        Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
```

```
    ElseIf (mButton = MOUSE_MIDDLE) Then
```

```
        Call mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
```

```

    Call mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
Else
    Call mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
    Call mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
End If
End Sub

```

```

Sub SendMausDoubleClick(ByVal mButton As Long)
SendMouseClick (mButton)
SendMouseClick (mButton)
End Sub

```

```

'-----Window

```

```

' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel
'
' sTitel: muss nicht der exakte Fenstertitel sein
'       hier kann bspw. auch nur der Anfang des Fenstertitel
'       angegeben werden, z.B.: Fenstertitel*
'

```

```

Public Function FindWindowHandle(ByVal sTitle As String) As Long
    Dim lngHwnd As Long
    Dim sText As String

```

```

    ' alle Fenster durchlaufen
    lngHwnd = FindWindow(vbNullString, vbNullString)
    Do While lngHwnd <> 0

```

```

        ' Fenstertitel ermitteln
        sText = GetWindowText(lngHwnd)
        Debug.Print lngHwnd & " " & sText
        If Len(sText) > 0 And LCase$(sText) Like LCase$(sTitle) Then
            FindWindowHandle = lngHwnd: Exit Do
        End If

```

```

        ' Nächstes Fenster
        lngHwnd = GetWindow(lngHwnd, GW_HWNDNEXT)
    Loop
End Function

```

```

' Hilfsfunktion zum Ermitteln des Fenstertitels

```

```

Public Function GetWindowText(ByVal hwnd As Long) As String
    Dim lResult As Long
    Dim sTemp As String

```

```

    lResult = GetWindowTextLength(hwnd) + 1
    sTemp = Space(lResult)
    lResult = GetWindowText(hwnd, sTemp, lResult)
    GetWindowText = left(sTemp, Len(sTemp) - 1)
End Function

```

```

' Module Name: ModFindWindowLike
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)

```

' Written 02/06/2005

Public Function FnFindWindowLike(strWindowTitle As String) As Long

'We'll pass a custom structure in as the parameter to store our result...

Dim Parameters As FindWindowParameters

Parameters.strTitle = strWindowTitle ' Input parameter

Call EnumWindows(AddressOf EnumWindowProc, VarPtr(Parameters))

FnFindWindowLike = Parameters.hwnd

End Function

Private Function EnumWindowProc(ByVal hwnd As Long, \_  
IParam As FindWindowParameters) As Long

Dim strWindowTitle As String

strWindowTitle = Space(260)

Call GetWindowText(hwnd, strWindowTitle, 260)

strWindowTitle = TrimNull(strWindowTitle) ' Remove extra null terminator

If strWindowTitle Like IParam.strTitle Then

IParam.hwnd = hwnd 'Store the result for later.

EnumWindowProc = 0 'This will stop enumerating more windows

Else

EnumWindowProc = 1

End If

End Function

' Module Name: ModSetForegroundWindow

' (c) 2005 Wayne Phillips (<http://www.everythingaccess.com>)

' Written 02/06/2005

Private Function TrimNull(strNullTerminatedString As String)

Dim lngPos As Long

'Remove unnecessary null terminator

lngPos = InStr(strNullTerminatedString, Chr\$(0))

If lngPos Then

TrimNull = left\$(strNullTerminatedString, lngPos - 1)

Else

TrimNull = strNullTerminatedString

End If

End Function



```
Public Function FnSetForegroundWindow(strWindowTitle As String) As Boolean
```

```
    Dim MyAppHWnd As Long  
    Dim CurrentForegroundThreadID As Long  
    Dim NewForegroundThreadID As Long  
    Dim lngRetVal As Long
```

```
    Dim blnSuccessful As Boolean
```

```
    MyAppHWnd = FnFindWindowLike(strWindowTitle)
```

```
    If MyAppHWnd <> 0 Then
```

```
        'We've found the application window by the caption  
        CurrentForegroundThreadID = GetWindowThreadProcessId(GetForegroundWindow(), ByVal  
0&)
```

```
        NewForegroundThreadID = GetWindowThreadProcessId(MyAppHWnd, ByVal 0&)
```

```
        'AttachThreadInput is used to ensure SetForegroundWindow will work
```

```
        'even if our application isn't currently the foreground window
```

```
        '(e.g. an automated app running in the background)
```

```
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, True)
```

```
        lngRetVal = SetForegroundWindow(MyAppHWnd)
```

```
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, False)
```

```
    If lngRetVal <> 0 Then
```

```
        'Now that the window is active, let's restore it from the taskbar
```

```
        If IsIconic(MyAppHWnd) Then
```

```
            Call ShowWindow(MyAppHWnd, SW_RESTORE)
```

```
        Else
```

```
            Call ShowWindow(MyAppHWnd, SW_SHOW)
```

```
        End If
```

```
        blnSuccessful = True
```

```
    Else
```

```
        MsgBox "Found the window, but failed to bring it to the foreground!"
```

```
    End If
```

```
Else
```

```
    'Failed to find the window caption
```

```
    'Therefore the app is probably closed.
```

```
    MsgBox "Application Window '" + strWindowTitle + "' not found!"
```

```
End If
```

```
FnSetForegroundWindow = blnSuccessful
```

End Function

```
'-----Microsoft Internet Controls (Tools>References: Microsoft Internet Controls)
'returns new instance of Internet Explorer
Function GetNewIE() As SHDocVw.InternetExplorer
    'create new IE instance
    Set GetNewIE = New SHDocVw.InternetExplorer
    'start with a blank page
    GetNewIE.Navigate2 "about:Blank"
End Function
```

```
'loads a web page and returns True or False depending on
'whether the page could be loaded or not
Function LoadWebPage(i_IE As SHDocVw.InternetExplorer, _
    i_URL As String) As Boolean
    With i_IE
        'open page
        .Navigate i_URL
        'wait until IE finished loading the page
        Do While .ReadyState <> READYSTATE_COMPLETE
            Application.Wait Now + TimeValue("0:00:01")
        Loop
        'check if page could be loaded
        If .Document.URL = i_URL Then
            LoadWebPage = True
        End If
    End With
End Function
```

```
'finds an open IE site by checking the URL
Function GetOpenIEByURL(ByVal i_URL As String) _
    As SHDocVw.InternetExplorer
```

```
Dim objShellWindows As New SHDocVw.ShellWindows
```

```
'ignore errors when accessing the document property
On Error Resume Next
'loop over all Shell-Windows
For Each GetOpenIEByURL In objShellWindows
    'if the document is of type HTMLDocument, it is an IE window
    If TypeName(GetOpenIEByURL.Document) = "HTMLDocument" Then
        'check the URL
        If GetOpenIEByURL.Document.URL = i_URL Then
            'leave, we found the right window
            Exit Function
        End If
    End If
Next
End Function
```

```
'finds an open IE site by checking the title
```

```
Function GetOpenIEByTitle(i_Title As String, _  
    Optional ByVal i_ExactMatch As Boolean = True) _  
    As SHDocVw.InternetExplorer
```

```
Dim objShellWindows As New SHDocVw.ShellWindows
```

```
If i_ExactMatch = False Then i_Title = "*" & i_Title & "*"  
'ignore errors when accessing the document property  
On Error Resume Next  
'loop over all Shell-Windows  
For Each GetOpenIEByTitle In objShellWindows  
'if the document is of type HTMLDocument, it is an IE window  
If TypeName(GetOpenIEByTitle.Document) = "HTMLDocument" Then  
'check the title  
If GetOpenIEByTitle.Document.Title Like i_Title Then  
'leave, we found the right window  
Exit Function  
End If  
End If  
Next  
End Function
```

```
Function IE_Preparation(myPageTitle As String, myPageURL As String) _  
    As SHDocVw.InternetExplorer
```

```
'IE object instantiation  
Dim myIE As SHDocVw.InternetExplorer
```

```
'check if page is already open  
Set myIE = GetOpenIEByTitle(myPageTitle, False)
```

```
'check if page is already open  
Set myIE = GetOpenIEByURL(myPageURL)
```

```
'if the page is not open then try to open it  
If myIE Is Nothing Then  
'page isn't open yet  
'create new IE instance  
Set myIE = GetNewIE  
'make IE window visible  
myIE.Visible = True  
'load page  
If LoadWebPage(myIE, myPageURL) = False Then  
'page wasn't loaded  
MsgBox "Couldn't open page"  
Exit Function  
End If  
End If
```

```
'move and resize the window  
Dim nhWnd As Long  
'nhWnd1 = FindWindow(vbNullString, "SmartChoice - Windows Internet Explorer provided by  
Syngenta")
```

```
nhWnd1 = myIE.hwnd
```

```
If nhWnd1 <> 0 Then
```

```
    MoveWindow nhWnd1, 0, 0, 1200, 900, 1
```

```
End If
```

```
Set IE_Preparation = myIE
```

```
End Function
```

```
Sub IE_CreateReport()
```

```
'SmC IE page title (used to check if already open)
```

```
Const myPageTitle As String = "SmartChoice"
```

```
'SmC stage (used to open the site)
```

```
'Const myPageURL As String =
```

```
"http://opxscs.eame.syngenta.org/STAGE/OPX2/frgolappschs02.eame.syngenta.org:8404/HOME?  
dispatched=http://opxscs.eame.syngenta.org/STAGE/OPX2/15.141.4.48:8100/"
```

```
'SmC prod (used to open the site)
```

```
Const myPageURL As String =
```

```
"http://opxscp.eame.syngenta.org/PROD/OPX2/frgolappschp01.eame.syngenta.org:8401/HOME?  
dispatched=http://opxscp.eame.syngenta.org/PROD/OPX2/15.141.4.50:8100/"
```

```
'minimize excel
```

```
Dim MyXLhWnd As Long
```

```
MyXLhWnd = FindWindow("XLMAIN", vbNullString)
```

```
retVal = ShowWindow(hwnd, SW_MINIMIZED)
```

```
'IE object instantiation
```

```
Dim myIE As SHDocVw.InternetExplorer
```

```
'prepare the SmC window in an IE object
```

```
Set myIE = IE_Preparation(myPageTitle, myPageURL)
```

```
'get the IE handle
```

```
Dim MyIEhWnd As Long
```

```
MyIEhWnd = myIE.hwnd
```

```
'show or restore IE depending
```

```
If IsIconic(MyIEhWnd) Then
```

```
    Call ShowWindow(MyIEhWnd, SW_RESTORE)
```

```
Else
```

```
    Call ShowWindow(MyIEhWnd, SW_SHOW)
```

```
End If
```

```
'bring SmC IE to the foreground
```

```
SetForegroundWindow MyAppHWND
```

```
'IE 7 make sure you have only one tab open
```

```
'or find a solution to toggle between the tabs (send key Ctrl+Tab) until you found the SmC
```

```
'wait SmC to start
```

```
Application.Wait (Now + TimeValue("0:00:10"))
```

```
'communicate with SmC
```

```
'-----
```

```
Dim myAktuellPos As POINTAPI
```

```
Dim myZielPos As POINTAPI
```

```
Dim Ziel_X, Ziel_Y As Long
```

```
Dim Aktuell_X, Aktuell_Y As Long
```

```
'Open the module button
```

```
Ziel_X0 = 21 '(calibrate here)
```

```
Ziel_Y0 = 138 '(calibrate here)
```

```
SetCursorPos Ziel_X0, Ziel_Y0
```

```
Application.Wait (Now + TimeValue("0:00:01"))
```

```
'send mouse click
```

```
SendMouseClick (MOUSE_LEFT)
```

```
Application.Wait (Now + TimeValue("0:00:01"))
```

```
'Choose the module project
```

```
'new position
```

```
Ziel_X = Ziel_X0
```

```
Ziel_Y = Ziel_Y0 + 57
```

```
SetCursorPos Ziel_X, Ziel_Y
```

```
Application.Wait (Now + TimeValue("0:00:01"))
```

```
'send mouse click
```

```
SendMouseClick (MOUSE_LEFT)
```

```
Application.Wait (Now + TimeValue("0:00:15"))
```

```
'loop over all virtual portfolios
```

```
Dim VirtualPortfolio_DropdownPos As Variant
```

```
VirtualPortfolio_DropdownPos = Array(79, 95) ',111, 127, 143, 159)
```

```
For VirtualPortfolio_DropdownPos_Iter = LBound(VirtualPortfolio_DropdownPos) To _  
    UBound(VirtualPortfolio_DropdownPos)
```

```
    'show or restore IE depending
```

```
    If IsIconic(MyIEhWnd) Then
```

```
        Call ShowWindow(MyIEhWnd, SW_RESTORE)
```

```
    Else
```

```
        Call ShowWindow(MyIEhWnd, SW_SHOW)
```

```
    End If
```

```
'bring SmC IE to the foreground
```

```
SetForegroundWindow MyIEhWnd
```

```
'Open the virtual portfolio dropdown
```

```
'new position
```

```
Ziel_X = Ziel_X0 + 550
```

```
Ziel_Y = Ziel_Y0 + 62
```

```
SetCursorPos Ziel_X, Ziel_Y
```

```
Application.Wait (Now + TimeValue("0:00:01"))
```

```
'send mouse click
```

```
SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:01"))
```

```
'Choose a virtual portfolio
'new position
Ziel_X = Ziel_X0 + 420
Ziel_Y = Ziel_Y0 + VirtualPortfolio_DropdownPos(VirtualPortfolio_DropdownPos_Iter)
SetCursorPos Ziel_X, Ziel_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:10"))
```

```
'Click the select all button
'new position
Ziel_X = Ziel_X0
Ziel_Y = Ziel_Y0 + 150
SetCursorPos Ziel_X, Ziel_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:01"))
'wait SmC to select the projects
Application.Wait (Now + TimeValue("0:00:10"))
```

```
'click the download into excel button
'new position
Ziel_X = Ziel_X0 + 1019
Ziel_Y = Ziel_Y0
SetCursorPos Ziel_X, Ziel_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:05"))
```

```
'Make IE, which downloads the projects, active/the foreground window
strWindowTitle = "Windows Internet Explorer provided by Syngenta"
Dim MyIEXLhWnd As String
MyIEXLhWnd = FindWindowHandle(strWindowTitle)
If MyIEXLhWnd <> 0 Then
    ' Fenster aktivieren und in den Vordergrund holen
    SetForegroundWindow MyIEXLhWnd
End If
Application.Wait (Now + TimeValue("0:00:03"))
```

```
'Click the open button in the IE msgbox
'new position
Ziel_X = Ziel_X0 + 586
Ziel_Y = Ziel_Y0 + 372
SetCursorPos Ziel_X, Ziel_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClicked (MOUSE_LEFT)
```

```

Application.Wait (Now + TimeValue("0:00:05"))

'minimize IE
retVal = ShowWindow(MyIEXLhWnd, SW_MINIMIZED)
Application.Wait (Now + TimeValue("0:00:01"))

'close IE
'lngReturnValue = PostMessage(MyIEXLhWnd, WM_CLOSE, 0&, 0&)

'make excel active/the foreground window (msgbox gets sometimes killed)
'Application.DisplayAlerts = False
'AppActivate "Microsoft Excel"
strWindowTitle = "Microsoft Excel"
MyXLhWnd = FindWindowHandle(strWindowTitle)
If MyXLhWnd <> 0 Then
    ' Fenster aktivieren und in den Vordergrund holen
    SetForegroundWindow MyXLhWnd
End If
Application.Wait (Now + TimeValue("0:00:03"))
'Application.DisplayAlerts = True

'(msgbox gets killed)
Ziel_X = Ziel_X0 + 567
Ziel_Y = Ziel_Y0 + 552
SetCursorPos Ziel_X, Ziel_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:05"))

'minimize excel
'With GetObject(, "Microsoft Excel")
'.ActiveWindow.WindowState = 1 ' olMinimized = 1
'End With
MyXLhWnd = FindWindow("XLMAIN", vbNullString)
retVal = ShowWindow(MyXLhWnd, SW_MINIMIZED)
Application.Wait (Now + TimeValue("0:00:01"))

'Stop
Next VirtualPortfolio_DropdownPos_Iter
Stop
End Sub

Sub determineMousePos()
Dim myAktuellPos As POINTAPI
Dim myZielPos As POINTAPI
Dim Ziel_X, Ziel_Y As Long
Dim Aktuell_X, Aktuell_Y As Long
'aktuelle pos abfragen
'If necessary change to VBA with ALT-Tab and start with F5 or F8
Do While 1
GetCursorPos myAktuellPos

```

```

Aktuell_X = myAktuellPos.x
Aktuell_Y = myAktuellPos.y
Debug.Print Aktuell_X & " " & Aktuell_Y
Stop
Loop
End Sub

```

```

Sub test()
Application.Wait (Now + TimeValue("0:00:02"))
Ziel_X0 = 21 '(calibrate here)
Ziel_Y0 = 138 '(calibrate here)
'download into excel
'new position
Ziel_X = Ziel_X0 + 1019
Ziel_Y = Ziel_Y0
SetCursorPos Ziel_X, Ziel_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:10"))
'Open
strWindowTitle = "Windows Internet Explorer provided by Syngenta"
hwnd = FindWindowHandle(strWindowTitle)
If hwnd <> 0 Then
    ' Fenster aktivieren und in den Vordergrund holen
    SetForegroundWindow hwnd
End If
'new position
Ziel_X = Ziel_X0 + 586
Ziel_Y = Ziel_Y0 + 372
SetCursorPos Ziel_X, Ziel_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:01"))

```

```

strWindowTitle = "Microsoft Excel"
hwnd = FindWindowHandle(strWindowTitle)
If hwnd <> 0 Then
    ' Fenster aktivieren und in den Vordergrund holen
    SetForegroundWindow hwnd
End If

```

```

'Ziel_X = Ziel_X0 + 567
'Ziel_Y = Ziel_Y0 + 552
'SetCursorPos Ziel_X, Ziel_Y
'Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
'SendMouseClick (MOUSE_LEFT)
'Application.Wait (Now + TimeValue("0:00:01"))
'Stop

```



End Sub

'-----Mouse

```
Private Declare Function SetCursorPos Lib "user32.dll" ( _  
    ByVal x As Long, _  
    ByVal y As Long) As Long
```

```
Private Declare Sub Sleep Lib "kernel32.dll" ( _  
    ByVal dwMilliseconds As Long)
```

```
Private Declare Function GetCursorPos Lib "user32.dll" ( _  
    ByRef lpPoint As POINTAPI) As Long
```

Private Type POINTAPI

    x As Long

    y As Long

End Type

```
Private Declare Sub mouse_event Lib "user32" _  
    (ByVal dwFlags As Long, ByVal dx As Long, _  
    ByVal dy As Long, ByVal cButtons As Long, _  
    ByVal dwExtraInfo As Long)
```

Public Const MOUSE\_LEFT = 0

Public Const MOUSE\_MIDDLE = 1

Public Const MOUSE\_RIGHT = 2

'-----keyboard

```
Private Declare Sub keybd_event Lib "user32.dll" (ByVal bVk As Byte, ByVal bScan As Byte, ByVal  
dwFlags As Long, _  
ByVal dwExtraInfo As Long)
```

Const VK\_STARTKEY = &H5B

Const VK\_M = 77

Const KEYEVENTF\_KEYUP = &H2

'-----Window

```
Private Declare Function PostMessage Lib "user32" _  
    Alias "PostMessageA" _  
    (ByVal hwnd As Long, _  
    ByVal wMsg As Long, _  
    ByVal wParam As Long, _  
    ByVal lParam As Long) As Long
```

Private Const WM\_CLOSE = &H10

```
Declare Function MoveWindow Lib "user32.dll" ( _  
    ByVal hwnd As Long, _  
    ByVal x As Long, _  
    ByVal y As Long, _  
    ByVal nWidth As Long, _  
    ByVal nHeight As Long, _
```

ByVal bRepaint As Long) As Long

```
Private Declare Function FindWindow Lib "user32" _  
    Alias "FindWindowA" ( _  
        ByVal lpClassName As String, _  
        ByVal lpWindowName As String) As Long
```

```
' Module Name: ModFindWindowLike  
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)  
' Written 02/06/2005
```

```
Private Declare Function EnumWindows Lib "user32" _  
    (ByVal lpEnumFunc As Long, _  
     ByVal lParam As Long) As Long
```

```
Private Declare Function GetWindowText Lib "user32" _  
    Alias "GetWindowTextA" _  
    (ByVal hwnd As Long, _  
     ByVal lpString As String, _  
     ByVal cch As Long) As Long
```

```
' Custom structure for passing in the parameters in/out of the hook enumeration function  
' Could use global variables instead, but this is nicer.
```

```
Private Type FindWindowParameters
```

```
    strTitle As String 'INPUT  
    hwnd As Long      'OUTPUT
```

```
End Type
```

```
' Module Name: Modz_SetForegroundWindow  
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)  
' Written 02/06/2005
```

```
Private Declare Function SetForegroundWindow Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Private Declare Function GetWindowThreadProcessId Lib "user32" _  
    (ByVal hwnd As Long, _  
     lpdwProcessId As Long) As Long
```

```
Private Declare Function IsIconic Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Private Declare Function ShowWindow Lib "user32" _  
    (ByVal hwnd As Long, _  
     ByVal nCmdShow As Long) As Long
```

```
Private Declare Function AttachThreadInput Lib "user32" _  
    (ByVal idAttach As Long, _  
     ByVal idAttachTo As Long, _  
     ByVal fAttach As Long) As Long
```

```
Private Declare Function GetForegroundWindow Lib "user32" _
    () As Long
```

```
Private Const SW_RESTORE = 9
```

```
Private Const SW_SHOW = 5
```

```
' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel
```

```
,
```

```
' sTitel: muss nicht der exakte Fenstertitel sein
```

```
'     hier kann bspw. auch nur der Anfang des Fenstertitel
```

```
'     angegeben werden, z.B.: Fenstertitel*
```

```
,
```

```
' benötigte API-Deklarationen
```

```
Private Declare Function GetWindowTextLength Lib "user32" _
```

```
    Alias "GetWindowTextLengthA" ( _
```

```
    ByVal hwnd As Long) As Long
```

```
Private Declare Function GetWindow Lib "user32" ( _
```

```
    ByVal hwnd As Long, _
```

```
    ByVal wCmd As Long) As Long
```

```
Private Const GW_HWNDNEXT = 2
```

```
'-----Mouse
```

```
'Die nachfolgende Prozedur simuliert den gewünschten Mausklick.
```

```
Public Sub SendMouseClicked(ByVal mButton As Long)
```

```
    Const MOUSEEVENTF_LEFTDOWN = &H2
```

```
    Const MOUSEEVENTF_LEFTUP = &H4
```

```
    Const MOUSEEVENTF_MIDDLEDOWN = &H20
```

```
    Const MOUSEEVENTF_MIDDLEUP = &H40
```

```
    Const MOUSEEVENTF_RIGHTDOWN = &H8
```

```
    Const MOUSEEVENTF_RIGHTUP = &H10
```

```
    If (mButton = MOUSE_LEFT) Then
```

```
        Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
```

```
        Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
```

```
    ElseIf (mButton = MOUSE_MIDDLE) Then
```

```
        Call mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
```

```
        Call mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
```

```
    Else
```

```
        Call mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
```

```
        Call mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
```

```
    End If
```

```
End Sub
```

```
Sub SendMausDoubleClick(ByVal mButton As Long)
```

```
    SendMouseClicked (mButton)
```

```
SendMouseClicked (mButton)
End Sub
```

```
'-----keyobard
Function z_ShowDesktop()
'Keyboard: Windows button + M button shows the desktop
'Do not test with debug F8 -> VBA Windows hides!!!
'http://msdn.microsoft.com/en-us/library/ms646304(VS.85).aspx
'http://msdn.microsoft.com/en-us/library/dd375731(v=VS.85).aspx

'WinKey down
keybd_event VK_STARTKEY, 0, 0, 0
'M key down
keybd_event VK_M, 0, 0, 0
'M key up
keybd_event VK_M, 0, KEYEVENTF_KEYUP, 0
'WinKey up
keybd_event VK_STARTKEY, 0, KEYEVENTF_KEYUP, 0

'do not minimiz form itself
'Me.WindowState = vbMaximized
'Me.WindowState = vbNormal[/b]
End Function
```

```
'-----Window

' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel
,

' sTitel: muss nicht der exakte Fenstertitel sein
' hier kann bspw. auch nur der Anfang des Fenstertitel
' angegeben werden, z.B.: Fenstertitel*
,

Public Function FindWindowHandle(ByVal sTitle As String) As Long
Dim lngHwnd As Long
Dim sText As String

' alle Fenster durchlaufen
lngHwnd = FindWindow(vbNullString, vbNullString)
Do While lngHwnd <> 0

' Fenstertitel ermitteln
sText = GetWindowText(lngHwnd)
Debug.Print lngHwnd & " " & sText
If Len(sText) > 0 And LCase$(sText) Like LCase$(sTitle) Then
FindWindowHandle = lngHwnd: Exit Do
End If

' Nächstes Fenster
lngHwnd = GetWindow(lngHwnd, GW_HWNDNEXT)
Loop
End Function
```

```

' Hilfsfunktion zum Ermitteln des Fenstertitels
Public Function GetWindowTitle(ByVal hwnd As Long) As String
    Dim lResult As Long
    Dim sTemp As String

    lResult = GetWindowTextLength(hwnd) + 1
    sTemp = Space(lResult)
    lResult = GetWindowText(hwnd, sTemp, lResult)
    GetWindowTitle = left(sTemp, Len(sTemp) - 1)
End Function

```

```

' Module Name: ModFindWindowLike
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)
' Written 02/06/2005
Public Function FnFindWindowLike(strWindowTitle As String) As Long

    'We'll pass a custom structure in as the parameter to store our result...
    Dim Parameters As FindWindowParameters
    Parameters.strTitle = strWindowTitle ' Input parameter

    Call EnumWindows(AddressOf EnumWindowProc, VarPtr(Parameters))

    FnFindWindowLike = Parameters.hwnd

End Function

```

```

Private Function EnumWindowProc(ByVal hwnd As Long, _
                                lParam As FindWindowParameters) As Long

    Dim strWindowTitle As String

    strWindowTitle = Space(260)
    Call GetWindowText(hwnd, strWindowTitle, 260)
    strWindowTitle = TrimNull(strWindowTitle) ' Remove extra null terminator

    If strWindowTitle Like lParam.strTitle Then

        lParam.hwnd = hwnd 'Store the result for later.
        EnumWindowProc = 0 'This will stop enumerating more windows

    Else

        EnumWindowProc = 1

    End If

End Function

```

```

' Module Name: ModSetForegroundWindow
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)
' Written 02/06/2005

```

```
Private Function TrimNull(strNullTerminatedString As String)
```

```
    Dim lngPos As Long
```

```
    'Remove unnecessary null terminator
```

```
    lngPos = InStr(strNullTerminatedString, Chr$(0))
```

```
    If lngPos Then
```

```
        TrimNull = Left$(strNullTerminatedString, lngPos - 1)
```

```
    Else
```

```
        TrimNull = strNullTerminatedString
```

```
    End If
```

```
End Function
```

```
Public Function z_SetForegroundWindow(strWindowTitle As String) As Boolean
```

```
    Dim MyAppHwnd As Long
```

```
    Dim CurrentForegroundThreadID As Long
```

```
    Dim NewForegroundThreadID As Long
```

```
    Dim lngRetVal As Long
```

```
    Dim blnSuccessful As Boolean
```

```
    MyAppHwnd = FnFindWindowLike(strWindowTitle)
```

```
    If MyAppHwnd <> 0 Then
```

```
        'We've found the application window by the caption
```

```
        CurrentForegroundThreadID = GetWindowThreadProcessId(GetForegroundWindow(), ByVal 0&)
```

```
        NewForegroundThreadID = GetWindowThreadProcessId(MyAppHwnd, ByVal 0&)
```

```
        'AttachThreadInput is used to ensure SetForegroundWindow will work
```

```
        'even if our application isn't currently the foreground window
```

```
        '(e.g. an automated app running in the background)
```

```
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, True)
```

```
        lngRetVal = SetForegroundWindow(MyAppHwnd)
```

```
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, False)
```

```
    If lngRetVal <> 0 Then
```

```
        'Now that the window is active, let's restore it from the taskbar
```

```
        If IsIconic(MyAppHwnd) Then
```

```
            Call ShowWindow(MyAppHwnd, SW_RESTORE)
```

```
        Else
```

```
            Call ShowWindow(MyAppHwnd, SW_SHOW)
```

```
        End If
```

```
        blnSuccessful = True
```

```
    Else
```

```
        MsgBox "Found the window, but failed to bring it to the foreground!"
```

```
    End If
```

```
Else
```

```
    'Failed to find the window caption
```

```
    'Therefore the app is probably closed.
```

```
    MsgBox "Application Window '" + strWindowTitle + "' not found!"
```

```
End If
```

```
    z_SetForegroundWindow = blnSuccessful
```

```
End Function
```

```
Public Function z_SetForegroundWindow2(MyAppHwnd As Long) As Boolean
```

```
    Dim CurrentForegroundThreadID As Long
```

```

Dim NewForegroundThreadID As Long
Dim lngRetVal As Long
Dim blnSuccessful As Boolean
If MyAppHWND <> 0 Then
    'We've found the application window by the caption
    CurrentForegroundThreadID = GetWindowThreadProcessId(GetForegroundWindow(), ByVal 0&)
    NewForegroundThreadID = GetWindowThreadProcessId(MyAppHWND, ByVal 0&)
    'AttachThreadInput is used to ensure SetForegroundWindow will work
    'even if our application isn't currently the foreground window
    '(e.g. an automated app running in the background)
    Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, True)
    lngRetVal = SetForegroundWindow(MyAppHWND)
    Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, False)
    If lngRetVal <> 0 Then
        'Now that the window is active, let's restore it from the taskbar
        If IsIconic(MyAppHWND) Then
            Call ShowWindow(MyAppHWND, SW_RESTORE)
        Else
            Call ShowWindow(MyAppHWND, SW_SHOW)
        End If
        blnSuccessful = True
    Else
        MsgBox "Found the window, but failed to bring it to the foreground!"
    End If
Else
    'Failed to find the window caption
    'Therefore the app is probably closed.
    MsgBox "Application Window '" + strWindowTitle + "' not found!"
End If
z_SetForegroundWindow2 = blnSuccessful
End Function

```

```

'-----Microsoft Internet Controls (Tools>References: Microsoft Internet Controls)
'returns new instance of Internet Explorer
Function GetNewIE() As SHDocVw.InternetExplorer
    'create new IE instance
    Set GetNewIE = New SHDocVw.InternetExplorer
    'start with a blank page
    GetNewIE.Navigate2 "about:Blank"
End Function

```

```

'loads a web page and returns True or False depending on
'whether the page could be loaded or not
Function LoadWebPage(i_IE As SHDocVw.InternetExplorer, _
    i_URL As String) As Boolean
    With i_IE
        'open page
        .Navigate i_URL
        'wait until IE finished loading the page
        Do While .ReadyState <> READYSTATE_COMPLETE
            Application.Wait Now + TimeValue("0:00:01")
        Loop
    End With

```

```

'check if page could be loaded
If .Document.URL = i_URL Then
    LoadWebPage = True
End If
End With
End Function

```

```

'finds an open IE site by checking the URL
Function GetOpenIEByURL(ByVal i_URL As String) _
    As SHDocVw.InternetExplorer

```

```

Dim objShellWindows As New SHDocVw.ShellWindows

```

```

'ignore errors when accessing the document property
On Error Resume Next
'loop over all Shell-Windows
For Each GetOpenIEByURL In objShellWindows
    'if the document is of type HTMLDocument, it is an IE window
    If TypeName(GetOpenIEByURL.Document) = "HTMLDocument" Then
        'check the URL
        If GetOpenIEByURL.Document.URL = i_URL Then
            'leave, we found the right window
            Exit Function
        End If
    End If
Next
End Function

```

```

'finds an open IE site by checking the title
Function GetOpenIEByTitle(i_Title As String, _
    Optional ByVal i_ExactMatch As Boolean = True) _
    As SHDocVw.InternetExplorer

```

```

Dim objShellWindows As New SHDocVw.ShellWindows

```

```

If i_ExactMatch = False Then i_Title = "*" & i_Title & "*"
'ignore errors when accessing the document property
On Error Resume Next
'loop over all Shell-Windows
For Each GetOpenIEByTitle In objShellWindows
    'if the document is of type HTMLDocument, it is an IE window
    If TypeName(GetOpenIEByTitle.Document) = "HTMLDocument" Then
        'check the title
        If GetOpenIEByTitle.Document.Title Like i_Title Then
            'leave, we found the right window
            Exit Function
        End If
    End If
Next
End Function

```

```

Function IE_Preparation(myPageTitle As String, myPageURL As String) _
    As SHDocVw.InternetExplorer

```



```

'IE object instantiation
Dim myIE As SHDocVw.InternetExplorer

'check if page is already open
Set myIE = GetOpenIEByTitle(myPageTitle, False)

'check if page is already open
Set myIE = GetOpenIEByURL(myPageURL)

'if the page is not open then try to open it
If myIE Is Nothing Then
    'page isn't open yet
    'create new IE instance
    Set myIE = GetNewIE
    'make IE window visible
    myIE.Visible = True
    'IE move and resize
    Call IE_MoveAndResize(myIE, 0, 0, 1200, 900)
    Application.Wait (Now + TimeValue("0:00:05"))
    'load page
    If LoadWebPage(myIE, myPageURL) = False Then
        'page wasn't loaded
        MsgBox "Couldn't open page"
        Exit Function
    End If
End If

'move and resize the window (do it before you start the URL)
'Dim nhWnd As Long
'nhWnd1 = FindWindow(vbNullString, "SmartChoice - Windows Internet Explorer provided by
Syngenta")
'nhWnd1 = myIE.hWnd
'If nhWnd1 <> 0 Then
'    MoveWindow nhWnd1, 0, 0, 1200, 900, 1
'End If

Set IE_Preparation = myIE
End Function

Function IE_MoveAndResize(ByRef IE As SHDocVw.InternetExplorer, _
    top As Variant, left As Variant, _
    width As Variant, height As Variant)
'move and resize the window
Dim nhWnd As Long
nhWnd = IE.hwnd
If nhWnd <> 0 Then
    MoveWindow nhWnd, top, left, width, height, 1
End If

End Function

Function z_CloseIE(strWindowTitle As String)

```

```

Const WM_CLOSE = &H10
Dim hwnd As Long
hwnd = FindWindowHandle(strWindowTitle)
If hwnd <> 0 Then
    'bring to the foreground
    SetForegroundWindow hwnd
    'close the IE window
    PostMessage hwnd, WM_CLOSE, 0&, 0&
End If
End Function

```

```

Function z_CloseIE2(hwnd As Long)
Const WM_CLOSE = &H10
If hwnd <> 0 Then
    'bring to the foreground
    SetForegroundWindow hwnd
    'close the IE window
    PostMessage hwnd, WM_CLOSE, 0&, 0&
End If
End Function

```

```

Public Function z_OpenAndActivateWb(wbname As String, wbpas As String, ByRef Wb As
Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Objective: Open the Workbook RD.xlsb if not already open and activate it.
'look if wbName is existent in workbooks list
Dim i As Long
For i = Workbooks.Count To 1 Step -1
    If Workbooks(i).Name = wbname Then Exit For
Next
'if wbName is existent in workbooks list, then i<>0-> activate workbook
'if wbName is not existent then i=0-> open workbook, activate workbook
If i <> 0 Then
    Set Wb = GetObject(wbpas & wbname)
    Wb.Activate
Else
    Set Wb = Workbooks.Open(wbpas & wbname)
    Wb.Activate
End If
End Function

```

```

Function z_WorkbookNewOrOpenOrActivate(wbname As String, wbpas As String, ByRef Wb As
Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 19.10.2011
'Objective: Open the Workbook RD.xlsb if not already open and activate it.
'look if wbName is existent in workbooks list
Dim i As Long
For i = Workbooks.Count To 1 Step -1
    If Workbooks(i).Name = wbname Then Exit For
Next

```

```

'if wbName is existent in workbooks list, then i<>0-> activate workbook
'if wbName is not existent then i=0-> open workbook, activate workbook
If i <> 0 Then
    Set Wb = GetObject(wbpath & wbname)
    Wb.Activate
Else
    On Error GoTo NewWB
    Set Wb = Workbooks.Open(wbpath & wbname)
    Wb.Activate
    On Error GoTo 0
End If

```

Exit Function

NewWB:

```

Set Wb = Workbooks.Add
Dim FileFormatValue As Integer
If wbname <> Empty Then
    Select Case LCase(Right(wbname, Len(wbname) - InStrRev(wbname, ".", , 1)))
        Case "xls": FileFormatValue = 56
        Case "xlsx": FileFormatValue = 51
        Case "xlsm": FileFormatValue = 52
        Case "xlsb": FileFormatValue = 50
        Case Else: FileFormatValue = 0
    End Select
End If
Wb.SaveAs filename:=wbpath & wbname, FileFormat:=FileFormatValue

```

End Function

Function z\_ExcelSessionWindowMinimized(Optional ByRef Wb As Workbook)

```

'Fenster der Excel session
If Wb Is Nothing Then
    Application.WindowState = xlMinimized
Else
    Wb.Application.WindowState = xlMinimized
End If

```

End Function

Function z\_ExcelSessionWindowNormal(Optional ByRef Wb As Workbook)

```

'Fenster der Excel session
If Wb Is Nothing Then
    Application.WindowState = xlNormal
Else
    Wb.Application.WindowState = xlNormal
    Wb.Activate
End If

```

End Function

Sub z\_ExcelSessionWindowMoveAndResize(Optional ByRef Wb As Workbook, \_

```

    Optional top As Variant, Optional left As Variant, _
    Optional width As Variant, Optional height As Variant)

```

```

If Wb Is Nothing Then
    If top <> Empty Then
        Application.top = top
    End If
    If left <> Empty Then

```

```

        Application.left = left
    End If
    If width <> Empty Then
        Application.width = width
    End If
    If height <> Empty Then
        Application.height = height
    End If
Else
    If top <> Empty Then
        Wb.Application.top = CInt(top)
    End If
    If left <> Empty Then
        Wb.Application.left = CInt(left)
    End If
    If width <> Empty Then
        Wb.Application.width = CInt(width)
    End If
    If height <> Empty Then
        Wb.Application.height = CInt(height)
    End If
End If
End Sub

```

```

Function z_ExcelWorkbookWindowMinimized(ByRef Wb As Workbook)

```

```

    'Fenster innerhalb der Excel session (workbooks)
    If Windows(Wb.Name).Visible Then
        ActiveWindow.WindowState = xlMinimized
    End If
End Function

```

```

Function z_ExcelWorkbookWindowNormal(ByRef Wb As Workbook)

```

```

    'Fenster innerhalb der Excel session (workbooks)
    If Windows(Wb.Name).Visible Then
        ActiveWindow.WindowState = xlMaximized
    End If
End Function

```

```

Function z_ExcelWorkbookWindowMinimizeAll(ByRef Wb_ref As Workbook)

```

```

    Application.ScreenUpdating = False
    Dim Wb As Workbook
    For Each Wb In Wb_ref.Application.Workbooks ' Workbooks
        'only those with status visible
        If Windows(Wb.Name).Visible Then
            ActiveWindow.WindowState = xlMinimized
        End If
    Next
    Application.ScreenUpdating = True
End Function

```

```

Sub IE_CreateReport()

```

```

    'close all windows/ Show the desktop
    Call z_ShowDesktop
    Application.Wait (Now + TimeValue("0:00:01"))

```

```
'choose a report Pls or Activitys
Dim SmC_Report As String
SmC_Report = "Activity" 'or " " for Pls
```

```
'add, open or activate an Excel workbook (and session)
Dim mywb As Workbook
Dim wbpath As String
Dim wbname As String
wbpath = "C:\Users\t740698\Desktop\"
wbname = "SmC_Download" & "_" & VBA.DateTime.Day(Now()) & "_" & _
    VBA.DateTime.Month(Now()) & "_" & VBA.DateTime.Year(Now()) & ".xlsb"
'wbname = "SmC_Download.xlsb"
Call z_WorkbookNewOrOpenOrActivate(wbname, wbpath, mywb)
Application.Wait (Now + TimeValue("0:00:03"))
```

```
'move and resize excel session
Call z_ExcelSessionWindowNormal(mywb)
Call z_ExcelSessionWindowMoveAndResize(mywb, "0", "0", "700", "600")
Application.Wait (Now + TimeValue("0:00:03"))
```

```
'minimize all Excel workbooks within the Excel session
Call z_ExcelWorkbookWindowMinimizeAll(mywb)
```

```
'minimize excel session
'Dim MyXLhWnd As Long
'MyXLhWnd = FindWindow("XLMAIN", vbNullString)
'retVal = ShowWindow(hwnd, SW_MINIMIZED)
Call z_ExcelSessionWindowMinimized(mywb)
Application.Wait (Now + TimeValue("0:00:03"))
```

```
'prepare the SmC window in an IE object
'IE object declaration
Dim myIE As SHDocVw.InternetExplorer
Const myPageTitle As String = "SmartChoice"
Dim SmC_System As String
Dim myPageURL As String
```

```
'chose a SmC system
SmC_System = "Prod"
If SmC_System = "Prod" Then
    myPageURL = _
        "http://opxscp.eame.syngenta.org/PROD/OPX2/frgolappschp01.eame.syngenta.org:8401/HOME?
dispatched=http://opxscp.eame.syngenta.org/PROD/OPX2/15.141.4.50:8100/"
Else
    myPageURL = _
        "http://opxscs.eame.syngenta.org/STAGE/OPX2/frgolappschs02.eame.syngenta.org:8404/HOME?
dispatched=http://opxscs.eame.syngenta.org/STAGE/OPX2/15.141.4.48:8100/"
End If
```

```
'IE object instantiation
Set myIE = IE_Preparation(myPageTitle, myPageURL)
Application.Wait (Now + TimeValue("0:00:01"))
```

```

'get the IE handle
Dim My_IE_hWnd As Long
My_IE_hWnd = myIE.hwnd

'show or restore IE depending on its current state
If IsIconic(My_IE_hWnd) Then
    Call ShowWindow(My_IE_hWnd, SW_RESTORE)
Else
    Call ShowWindow(My_IE_hWnd, SW_SHOW)
End If

'bring SmC IE to the foreground
SetForegroundWindow My_IE_hWnd
Application.Wait (Now + TimeValue("0:00:01"))

'IE 7 make sure you have only one tab open!!!
'or find a solution to toggle between the tabs (send key Ctrl+Tab) until you found the SmC

'wait for SmC to start up
Application.Wait (Now + TimeValue("0:00:15"))

'communicate with SmC
'-----
Dim MyActuelPos As POINTAPI
Dim myTargetPos As POINTAPI
Dim Target_X, Target_Y As Long
Dim Actuel_X, Actuel_Y As Long

'Open the module button
Target_X0 = 21 '(calibrate here)
Target_Y0 = 138 '(calibrate here)
SetCursorPos Target_X0, Target_Y0
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:01"))

'Choose the module project
'new position
Target_X = Target_X0
Target_Y = Target_Y0 + 57
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:15"))

'loop over all virtual portfolios
Dim VirtualPortfolio_DropdownPos As Variant
VirtualPortfolio_DropdownPos = Array(79, 95) ',111, 127, 143, 159)

For VirtualPortfolio_DropdownPos_Iter = LBound(VirtualPortfolio_DropdownPos) To _

```

```

UBound(VirtualPortfolio_DropdownPos)

'show or restore IE depending
If IsIconic(MyIEhWnd) Then
    Call ShowWindow(My_IE_hWnd, SW_RESTORE)
Else
    Call ShowWindow(My_IE_hWnd, SW_SHOW)
End If

'bring SmC IE to the foreground
Call SetForegroundWindow(My_IE_hWnd)

'Open the virtual portfolio dropdown
'new position
Target_X = Target_X0 + 550
Target_Y = Target_Y0 + 62
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:01"))

'Choose a virtual portfolio
'new position
Target_X = Target_X0 + 420
Target_Y = Target_Y0 + VirtualPortfolio_DropdownPos(VirtualPortfolio_DropdownPos_Iter)
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:10"))

'Click the select all button
'new position
Target_X = Target_X0
Target_Y = Target_Y0 + 150
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:01"))
'wait SmC to select the projects
Application.Wait (Now + TimeValue("0:00:10"))

'open the activities or not depending on the requested report
If SmC_Report = "Activity" Then
    'click open

    'click the style

    '
Else
    'do nothing

```

End If

'click the download into excel button

'new position

Target\_X = Target\_X0 + 1019

Target\_Y = Target\_Y0

SetCursorPos Target\_X, Target\_Y

Application.Wait (Now + TimeValue("0:00:01"))

'send mouse click

SendMouseClick (MOUSE\_LEFT)

Application.Wait (Now + TimeValue("0:00:03"))

'Minimize SmC IE

'Call ShowWindow(My\_IE\_hWnd, SW\_MINIMIZE)

'Application.Wait (Now + TimeValue("0:00:01"))

'Make Prozess "File Download" active !!!!!

'Dim strWindowTitle As String

'strWindowTitle = "File Download"

'Dim My\_IE\_XL\_Download\_hWnd As Long

'My\_IE\_XL\_Download\_hWnd = FindWindowHandle(strWindowTitle)

'if the window could be found

'If My\_IE\_XL\_Download\_hWnd <> 0 Then

' 'bring to the foreground

' Call SetForegroundWindow(My\_IE\_XL\_Download\_hWnd)

' Application.Wait (Now + TimeValue("0:00:03"))

'End If

'Click the open button in the IE msgbox

'If My\_IE\_XL\_Download\_hWnd <> 0 Then

' 'new position

' Target\_X = Target\_X0 + 586

' Target\_Y = Target\_Y0 + 372

' SetCursorPos Target\_X, Target\_Y

' Application.Wait (Now + TimeValue("0:00:01"))

' 'send keyboard Alt+O

' 'Call z\_SetForegroundWindow2(My\_IE\_XL\_Download\_hWnd)

' 'Call Key\_Alt\_O

' 'send mouse click

' SendMouseClick (MOUSE\_LEFT)

' Application.Wait (Now + TimeValue("0:00:03"))

'End If

'minimize IE that prepares the download

'strWindowTitle = "Windows Internet Explorer provided by Syngenta"

Dim My\_IE\_XL\_hWnd As Long

'My\_IE\_XL\_hWnd = FindWindowHandle(strWindowTitle)

'If My\_IE\_XL\_hWnd <> 0 Then

' Call ShowWindow(My\_IE\_XL\_hWnd, SW\_MINIMIZE)

' Application.Wait (Now + TimeValue("0:00:03"))

'End If



```

'make excel active
Call z_ExcelSessionWindowNormal(mywb)
Application.Wait (Now + TimeValue("0:00:03"))

'if there pops up a msgbox click ok
Target_X = Target_X0 + 567
Target_Y = Target_Y0 + 552
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:05"))

'minimize all Excel workbooks within the Excel session
Call z_ExcelWorkbookWindowMinimizeAll(mywb)
Application.Wait (Now + TimeValue("0:00:01"))

'minimize excel
Call z_ExcelSessionWindowMinimized(mywb)
Application.Wait (Now + TimeValue("0:00:01"))

'If IE download fails, close IE
strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet Explorer
provided by Syngenta"
Dim My_IE_XL_Error_hWnd As Long
My_IE_XL_Error_hWnd = FindWindowHandle(strWindowTitle)
Call z_CloseIE2(My_IE_XL_Error_hWnd)
Application.Wait (Now + TimeValue("0:00:01"))

Stop
Next VirtualPortfolio_DropdownPos_Iter
Stop
End Sub

Sub determineMousePos()
Dim MyActuelPos As POINTAPI
Dim myTargetPos As POINTAPI
Dim Target_X, Target_Y As Long
Dim Actuel_X, Actuel_Y As Long
'aktuelle pos abfragen
'If necessary change to VBA with ALT-Tab and start with F5 or F8
Do While 1
GetCursorPos MyActuelPos
Actuel_X = MyActuelPos.x
Actuel_Y = MyActuelPos.y
Debug.Print Actuel_X & " " & Actuel_Y
Stop
Loop
End Sub

Function Key_Alt_O()
'http://www.unet.univie.ac.at/~a7425519/programme/hex2dez.htm

```

```

Const VK_MENU = 18 '0x12
Const VK_O = 79 '0x4F
'WinKey down
keybd_event VK_MENU, 0, 0, 0
'F key down
keybd_event VK_O, 0, 0, 0
'M key up
keybd_event VK_O, 0, KEYEVENTF_KEYUP, 0
'WinKey up
keybd_event VK_MENU, 0, KEYEVENTF_KEYUP, 0
End Function

```

```

'-----Mouse
Private Declare Function SetCursorPos Lib "user32.dll" ( _
    ByVal x As Long, _
    ByVal y As Long) As Long

Private Declare Sub Sleep Lib "kernel32.dll" ( _
    ByVal dwMilliseconds As Long)

Private Declare Function GetCursorPos Lib "user32.dll" ( _
    ByRef lpPoint As POINTAPI) As Long

```

```

Private Type POINTAPI
    x As Long
    y As Long
End Type

```

```

Private Declare Sub mouse_event Lib "user32" _
    (ByVal dwFlags As Long, ByVal dx As Long, _
    ByVal dy As Long, ByVal cButtons As Long, _
    ByVal dwExtraInfo As Long)

```

```

Public Const MOUSE_LEFT = 0
Public Const MOUSE_MIDDLE = 1
Public Const MOUSE_RIGHT = 2

```

```

'-----keyobard
Private Declare Sub keybd_event Lib "user32.dll" (ByVal bVk As Byte, ByVal bScan As Byte, ByVal
dwFlags As Long, _
ByVal dwExtraInfo As Long)

```

```

Const VK_STARTKEY = &H5B
Const VK_M = 77
Const KEYEVENTF_KEYUP = &H2

```

```

'-----Window
Private Declare Function PostMessage Lib "user32" _
    Alias "PostMessageA" _
    (ByVal hwnd As Long, _
    ByVal wMsg As Long, _

```

```
ByVal wParam As Long, _  
ByVal lParam As Long) As Long
```

```
Private Const WM_CLOSE = &H10
```

```
Declare Function MoveWindow Lib "user32.dll" ( _  
    ByVal hwnd As Long, _  
    ByVal x As Long, _  
    ByVal y As Long, _  
    ByVal nWidth As Long, _  
    ByVal nHeight As Long, _  
    ByVal bRepaint As Long) As Long
```

```
Private Declare Function FindWindow Lib "user32" _  
    Alias "FindWindowA" ( _  
    ByVal lpClassName As String, _  
    ByVal lpWindowName As String) As Long
```

```
' Module Name: ModFindWindowLike  
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)  
' Written 02/06/2005
```

```
Private Declare Function EnumWindows Lib "user32" _  
    (ByVal lpEnumFunc As Long, _  
    ByVal lParam As Long) As Long
```

```
Private Declare Function GetWindowText Lib "user32" _  
    Alias "GetWindowTextA" _  
    (ByVal hwnd As Long, _  
    ByVal lpString As String, _  
    ByVal cch As Long) As Long
```

```
' Custom structure for passing in the parameters in/out of the hook enumeration function  
' Could use global variables instead, but this is nicer.  
Private Type FindWindowParameters
```

```
    strTitle As String 'INPUT  
    hwnd As Long      'OUTPUT
```

```
End Type
```

```
' Module Name: Modz_SetForegroundWindow  
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)  
' Written 02/06/2005
```

```
Private Declare Function SetForegroundWindow Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Private Declare Function GetWindowThreadProcessId Lib "user32" _  
    (ByVal hwnd As Long, _  
    lpdwProcessId As Long) As Long
```

```
Private Declare Function IsIconic Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Private Declare Function ShowWindow Lib "user32" _  
    (ByVal hwnd As Long, _  
    ByVal nCmdShow As Long) As Long
```

```
Private Declare Function AttachThreadInput Lib "user32" _  
    (ByVal idAttach As Long, _  
    ByVal idAttachTo As Long, _  
    ByVal fAttach As Long) As Long
```

```
Private Declare Function GetForegroundWindow Lib "user32" _  
    () As Long
```

```
Private Const SW_RESTORE = 9  
Private Const SW_SHOW = 5
```

```
' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel  
,  
' sTitel: muss nicht der exakte Fenstertitel sein  
'     hier kann bspw. auch nur der Anfang des Fenstertitel  
'     angegeben werden, z.B.: Fenstertitel*  
,  
' benötigte API-Deklarationen
```

```
Private Declare Function GetWindowTextLength Lib "user32" _  
    Alias "GetWindowTextLengthA" ( _  
    ByVal hwnd As Long) As Long
```

```
Private Declare Function GetWindow Lib "user32" ( _  
    ByVal hwnd As Long, _  
    ByVal wCmd As Long) As Long
```

```
Private Const GW_HWNDNEXT = 2
```

```
'-----Mouse
```

```
'Die nachfolgende Prozedur simuliert den gewünschten Mausklick.
```

```
Public Sub SendMouseClicked(ByVal mButton As Long)  
    Const MOUSEEVENTF_LEFTDOWN = &H2  
    Const MOUSEEVENTF_LEFTUP = &H4  
    Const MOUSEEVENTF_MIDDLEDOWN = &H20  
    Const MOUSEEVENTF_MIDDLEUP = &H40  
    Const MOUSEEVENTF_RIGHTDOWN = &H8  
    Const MOUSEEVENTF_RIGHTUP = &H10
```

```
    If (mButton = MOUSE_LEFT) Then  
        Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)  
        Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
```

```

ElseIf (mButton = MOUSE_MIDDLE) Then
    Call mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
    Call mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
Else
    Call mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
    Call mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
End If
End Sub

```

```

Sub SendMausDoubleClick(ByVal mButton As Long)
SendMouseClicked (mButton)
SendMouseClicked (mButton)
End Sub

```

```

'-----keyobard
Function z_ShowDesktop()
'Keyboard: Windows button + M button shows the desktop
'Do not test with debug F8 -> VBA Windows hides!!!
'http://msdn.microsoft.com/en-us/library/ms646304(VS.85).aspx
'http://msdn.microsoft.com/en-us/library/dd375731(v=VS.85).aspx

```

```

'WinKey down
keybd_event VK_STARTKEY, 0, 0, 0
'M key down
keybd_event VK_M, 0, 0, 0
'M key up
keybd_event VK_M, 0, KEYEVENTF_KEYUP, 0
'WinKey up
keybd_event VK_STARTKEY, 0, KEYEVENTF_KEYUP, 0

```

```

'do not minimiz form itself
'Me.WindowState = vbMaximized
'Me.WindowState = vbNormal[/b]
End Function

```

```

'-----Window

```

```

' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel
'
' sTitel: muss nicht der exakte Fenstertitel sein
'       hier kann bspw. auch nur der Anfang des Fenstertitel
'       angegeben werden, z.B.: Fenstertitel*
'

```

```

Public Function FindWindowHandle(ByVal sTitle As String) As Long
Dim lngHWND As Long
Dim sText As String

```

```

' alle Fenster durchlaufen
lngHWND = FindWindow(vbNullString, vbNullString)
Do While lngHWND <> 0

```

```

' Fenstertitel ermitteln
sText = GetWindowText(IngHWnd)
Debug.Print IngHWnd & " " & sText
If Len(sText) > 0 And LCase$(sText) Like LCase$(sTitle) Then
    FindWindowHandle = IngHWnd: Exit Do
End If

' Nächstes Fenster
IngHWnd = GetWindow(IngHWnd, GW_HWNDNEXT)
Loop
End Function
' Hilfsfunktion zum Ermitteln des Fenstertitels
Public Function GetWindowText(ByVal hwnd As Long) As String
    Dim lResult As Long
    Dim sTemp As String

    lResult = GetWindowTextLength(hwnd) + 1
    sTemp = Space(lResult)
    lResult = GetWindowText(hwnd, sTemp, lResult)
    GetWindowText = left(sTemp, Len(sTemp) - 1)
End Function

' Module Name: ModFindWindowLike
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)
' Written 02/06/2005
Public Function FnFindWindowLike(strWindowTitle As String) As Long

    'We'll pass a custom structure in as the parameter to store our result...
    Dim Parameters As FindWindowParameters
    Parameters.strTitle = strWindowTitle ' Input parameter

    Call EnumWindows(AddressOf EnumWindowProc, VarPtr(Parameters))

    FnFindWindowLike = Parameters.hwnd
End Function

Private Function EnumWindowProc(ByVal hwnd As Long, _
    lParam As FindWindowParameters) As Long

    Dim strWindowTitle As String

    strWindowTitle = Space(260)
    Call GetWindowText(hwnd, strWindowTitle, 260)
    strWindowTitle = TrimNull(strWindowTitle) ' Remove extra null terminator

    If strWindowTitle Like lParam.strTitle Then

        lParam.hwnd = hwnd 'Store the result for later.
        EnumWindowProc = 0 'This will stop enumerating more windows

    Else

```

EnumWindowProc = 1

End If

End Function

' Module Name: ModSetForegroundWindow

' (c) 2005 Wayne Phillips (<http://www.everythingaccess.com>)

' Written 02/06/2005

Private Function TrimNull(strNullTerminatedString As String)

Dim lngPos As Long

'Remove unnecessary null terminator

lngPos = InStr(strNullTerminatedString, Chr\$(0))

If lngPos Then

TrimNull = Left\$(strNullTerminatedString, lngPos - 1)

Else

TrimNull = strNullTerminatedString

End If

End Function

Public Function z\_SetForegroundWindow(strWindowTitle As String) As Boolean

Dim MyAppHwnd As Long

Dim CurrentForegroundThreadId As Long

Dim NewForegroundThreadId As Long

Dim lngRetVal As Long

Dim blnSuccessful As Boolean

MyAppHwnd = FindWindowLike(strWindowTitle)

If MyAppHwnd <> 0 Then

'We've found the application window by the caption

CurrentForegroundThreadId = GetWindowThreadProcessId(GetForegroundWindow(), ByVal 0&)

NewForegroundThreadId = GetWindowThreadProcessId(MyAppHwnd, ByVal 0&)

'AttachThreadInput is used to ensure SetForegroundWindow will work

'even if our application isn't currently the foreground window

'(e.g. an automated app running in the background)

Call AttachThreadInput(CurrentForegroundThreadId, NewForegroundThreadId, True)

lngRetVal = SetForegroundWindow(MyAppHwnd)

Call AttachThreadInput(CurrentForegroundThreadId, NewForegroundThreadId, False)

If lngRetVal <> 0 Then

'Now that the window is active, let's restore it from the taskbar

If IsIconic(MyAppHwnd) Then

Call ShowWindow(MyAppHwnd, SW\_RESTORE)

Else

Call ShowWindow(MyAppHwnd, SW\_SHOW)

End If

blnSuccessful = True

Else

```

        MsgBox "Found the window, but failed to bring it to the foreground!"
    End If
Else
    'Failed to find the window caption
    'Therefore the app is probably closed.
    MsgBox "Application Window '" + strWindowTitle + "' not found!"
End If
z_SetForegroundWindow = blnSuccessful
End Function
Public Function z_SetForegroundWindow2(MyAppHWND As Long) As Boolean
    Dim CurrentForegroundThreadID As Long
    Dim NewForegroundThreadID As Long
    Dim lngRetVal As Long
    Dim blnSuccessful As Boolean
    If MyAppHWND <> 0 Then
        'We've found the application window by the caption
        CurrentForegroundThreadID = GetWindowThreadProcessId(GetForegroundWindow(), ByVal 0&)
        NewForegroundThreadID = GetWindowThreadProcessId(MyAppHWND, ByVal 0&)
        'AttachThreadInput is used to ensure SetForegroundWindow will work
        'even if our application isn't currently the foreground window
        '(e.g. an automated app running in the background)
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, True)
        lngRetVal = SetForegroundWindow(MyAppHWND)
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, False)
        If lngRetVal <> 0 Then
            'Now that the window is active, let's restore it from the taskbar
            If IsIconic(MyAppHWND) Then
                Call ShowWindow(MyAppHWND, SW_RESTORE)
            Else
                Call ShowWindow(MyAppHWND, SW_SHOW)
            End If
            blnSuccessful = True
        Else
            MsgBox "Found the window, but failed to bring it to the foreground!"
        End If
    Else
        'Failed to find the window caption
        'Therefore the app is probably closed.
        MsgBox "Application Window '" + strWindowTitle + "' not found!"
    End If
    z_SetForegroundWindow2 = blnSuccessful
End Function

```

```

'-----Microsoft Internet Controls (Tools>References: Microsoft Internet Controls)
'returns new instance of Internet Explorer
Function GetNewIE() As SHDocVw.InternetExplorer
    'create new IE instance
    Set GetNewIE = New SHDocVw.InternetExplorer
    'start with a blank page
    GetNewIE.Navigate2 "about:Blank"
End Function

```



'loads a web page and returns True or False depending on  
'whether the page could be loaded or not  
Function LoadWebPage(i\_IE As SHDocVw.InternetExplorer, \_  
i\_URL As String) As Boolean

With i\_IE  
'open page  
.Navigate i\_URL  
'wait until IE finished loading the page  
Do While .ReadyState <> READYSTATE\_COMPLETE  
Application.Wait Now + TimeValue("0:00:01")  
Loop  
'check if page could be loaded  
If .Document.URL = i\_URL Then  
LoadWebPage = True  
End If  
End With  
End Function

'finds an open IE site by checking the URL  
Function GetOpenIEByURL(ByVal i\_URL As String) \_  
As SHDocVw.InternetExplorer

Dim objShellWindows As New SHDocVw.ShellWindows

'ignore errors when accessing the document property  
On Error Resume Next  
'loop over all Shell-Windows  
For Each GetOpenIEByURL In objShellWindows  
'if the document is of type HTMLDocument, it is an IE window  
If TypeName(GetOpenIEByURL.Document) = "HTMLDocument" Then  
'check the URL  
If GetOpenIEByURL.Document.URL = i\_URL Then  
'leave, we found the right window  
Exit Function  
End If  
End If  
Next  
End Function

'finds an open IE site by checking the title  
Function GetOpenIEByTitle(i\_Title As String, \_  
Optional ByVal i\_ExactMatch As Boolean = True) \_  
As SHDocVw.InternetExplorer

Dim objShellWindows As New SHDocVw.ShellWindows

If i\_ExactMatch = False Then i\_Title = "\*" & i\_Title & "\*"  
'ignore errors when accessing the document property  
On Error Resume Next  
'loop over all Shell-Windows  
For Each GetOpenIEByTitle In objShellWindows  
'if the document is of type HTMLDocument, it is an IE window  
If TypeName(GetOpenIEByTitle.Document) = "HTMLDocument" Then

```

        'check the title
        If GetOpenIEByTitle.Document.Title Like i_Title Then
            'leave, we found the right window
            Exit Function
        End If
    End If
Next
End Function

Function IE_Preparation(myPageTitle As String, myPageURL As String) _
    As SHDocVw.InternetExplorer

    'IE object instantiation
    Dim myIE As SHDocVw.InternetExplorer

    'check if page is already open
    Set myIE = GetOpenIEByTitle(myPageTitle, False)

    'check if page is already open
    Set myIE = GetOpenIEByURL(myPageURL)

    'if the page is not open then try to open it
    If myIE Is Nothing Then
        'page isn't open yet
        'create new IE instance
        Set myIE = GetNewIE
        'make IE window visible
        myIE.Visible = True
        'IE move and resize
        Call IE_MoveAndResize(myIE, 0, 0, 1200, 900)
        Application.Wait (Now + TimeValue("0:00:05"))
        'load page
        If LoadWebPage(myIE, myPageURL) = False Then
            'page wasn't loaded
            MsgBox "Couldn't open page"
            Exit Function
        End If
    End If

    'move and resize the window (do it before you start the URL)
    'Dim nhWnd As Long
    'nhWnd1 = FindWindow(vbNullString, "SmartChoice - Windows Internet Explorer provided by Syngenta")
    'nhWnd1 = myIE.hWnd
    'If nhWnd1 <> 0 Then
    '    MoveWindow nhWnd1, 0, 0, 1200, 900, 1
    'End If

    Set IE_Preparation = myIE
End Function

Function IE_MoveAndResize(ByRef IE As SHDocVw.InternetExplorer, _
    top As Variant, left As Variant, _

```

```

        width As Variant, height As Variant)
'move and resize the window
Dim nhWnd As Long
nhWnd = IE.hwnd
If nhWnd <> 0 Then
    MoveWindow nhWnd, top, left, width, height, 1
End If

```

End Function

```

Function z_CloseIE(strWindowTitle As String)
    Const WM_CLOSE = &H10
    Dim hwnd As Long
    hwnd = FindWindowHandle(strWindowTitle)
    If hwnd <> 0 Then
        'bring to the foreground
        SetForegroundWindow hwnd
        'close the IE window
        PostMessage hwnd, WM_CLOSE, 0&, 0&
    End If
End Function

```

```

Function z_CloseIE2(hwnd As Long)
    Const WM_CLOSE = &H10
    If hwnd <> 0 Then
        'bring to the foreground
        SetForegroundWindow hwnd
        'close the IE window
        PostMessage hwnd, WM_CLOSE, 0&, 0&
    End If
End Function

```

```

Public Function z_OpenAndActivateWb(wbname As String, wbpath As String, ByRef Wb As
Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Objective: Open the Workbook RD.xlsb if not already open and activate it.
'look if wbName is existent in workbooks list
Dim i As Long
For i = Workbooks.Count To 1 Step -1
    If Workbooks(i).Name = wbname Then Exit For
Next
'if wbName is existent in workbooks list, then i<>0-> activate workbook
'if wbName is not existent then i=0-> open workbook, activate workbook
If i <> 0 Then
    Set Wb = GetObject(wbpath & wbname)
    Wb.Activate
Else
    Set Wb = Workbooks.Open(wbpath & wbname)
    Wb.Activate
End If
End Function

```

Function z\_WorkbookNewOrOpenOrActivate(wbname As String, wbpas As String, ByRef Wb As Workbook)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 19.10.2011

'Objective: Open the Workbook RD.xlsb if not already open and activate it.

'look if wbName is existent in workbooks list

Dim i As Long

For i = Workbooks.Count To 1 Step -1

    If Workbooks(i).Name = wbname Then Exit For

Next

'if wbName is existent in workbooks list, then i<>0-> activate workbook

'if wbName is not existent then i=0-> open workbook, activate workbook

If i <> 0 Then

    Set Wb = GetObject(wbpas & wbname)

    Wb.Activate

Else

    On Error GoTo NewWB

    Set Wb = Workbooks.Open(wbpas & wbname)

    Wb.Activate

    On Error GoTo 0

End If

Exit Function

NewWB:

Set Wb = Workbooks.Add

Dim FileFormatValue As Integer

If wbname <> Empty Then

    Select Case LCase(Right(wbname, Len(wbname) - InStrRev(wbname, ".", , 1)))

        Case "xls": FileFormatValue = 56

        Case "xlsx": FileFormatValue = 51

        Case "xlsm": FileFormatValue = 52

        Case "xlsb": FileFormatValue = 50

        Case Else: FileFormatValue = 0

    End Select

End If

Wb.SaveAs filename:=wbpas & wbname, FileFormat:=FileFormatValue

End Function

Function z\_ExcelSessionWindowMinimized(Optional ByRef Wb As Workbook)

'Fenster der Excel session

If Wb Is Nothing Then

    Application.WindowState = xlMinimized

Else

    Wb.Application.WindowState = xlMinimized

End If

End Function

Function z\_ExcelSessionWindowNormal(Optional ByRef Wb As Workbook)

'Fenster der Excel session

If Wb Is Nothing Then

    Application.WindowState = xlNormal

Else

    Wb.Application.WindowState = xlNormal

    Wb.Activate

End If  
End Function

```
Sub z_ExcelSessionWindowMoveAndResize(Optional ByRef Wb As Workbook, _  
    Optional top As Variant, Optional left As Variant, _  
    Optional width As Variant, Optional height As Variant)  
    If Wb Is Nothing Then  
        If top <> Empty Then  
            Application.top = top  
        End If  
        If left <> Empty Then  
            Application.left = left  
        End If  
        If width <> Empty Then  
            Application.width = width  
        End If  
        If height <> Empty Then  
            Application.height = height  
        End If  
    Else  
        If top <> Empty Then  
            Wb.Application.top = CInt(top)  
        End If  
        If left <> Empty Then  
            Wb.Application.left = CInt(left)  
        End If  
        If width <> Empty Then  
            Wb.Application.width = CInt(width)  
        End If  
        If height <> Empty Then  
            Wb.Application.height = CInt(height)  
        End If  
    End If  
End Sub
```

```
Function z_ExcelWorkbookWindowMinimized(ByRef Wb As Workbook)  
    'Fenster innerhalb der Excel session (workbooks)  
    If Windows(Wb.Name).Visible Then  
        ActiveWindow.WindowState = xlMinimized  
    End If  
End Function
```

```
Function z_ExcelWorkbookWindowNormal(ByRef Wb As Workbook)  
    'Fenster innerhalb der Excel session (workbooks)  
    If Windows(Wb.Name).Visible Then  
        ActiveWindow.WindowState = xlMaximized  
    End If  
End Function
```

```
Function z_ExcelWorkbookWindowMinimizeAll(ByRef Wb_ref As Workbook)  
    Application.ScreenUpdating = False  
    Dim Wb As Workbook  
    For Each Wb In Wb_ref.Application.Workbooks ' Workbooks  
        'only those with status visible  
        If Windows(Wb.Name).Visible Then
```

```

        ActiveWindow.WindowState = xlMinimized
    End If
Next
Application.ScreenUpdating = True
End Function

Sub IE_CreateReport()
'close all windows/ Show the desktop
Call z_ShowDesktop
Application.Wait (Now + TimeValue("0:00:01"))

'choose a report Pls or Activitys
Dim SmC_Report As String
SmC_Report = "Activity" 'or " " for Pls

'add, open or activate an Excel workbook (and session)
Dim mywb As Workbook
Dim wbpath As String
Dim wbname As String
wbpath = "C:\Users\t740698\Desktop\"
wbname = "SmC_Download" & "_" & VBA.DateTime.Day(Now()) & "_" & _
    VBA.DateTime.Month(Now()) & "_" & VBA.DateTime.Year(Now()) & ".xlsb"
'wbname = "SmC_Download.xlsb"
Call z_WorkbookNewOrOpenOrActivate(wbname, wbpath, mywb)
Application.Wait (Now + TimeValue("0:00:03"))

'move and resize excel session
Call z_ExcelSessionWindowNormal(mywb)
Call z_ExcelSessionWindowMoveAndResize(mywb, "0", "0", "700", "600")
Application.Wait (Now + TimeValue("0:00:03"))

'minimize all Excel workbooks within the Excel session
Call z_ExcelWorkbookWindowMinimizeAll(mywb)

'minimize excel session
'Dim MyXLhWnd As Long
'MyXLhWnd = FindWindow("XLMAIN", vbNullString)
'retVal = ShowWindow(hwnd, SW_MINIMIZED)
Call z_ExcelSessionWindowMinimized(mywb)
Application.Wait (Now + TimeValue("0:00:03"))

'prepare the SmC window in an IE object
'IE object declaration
Dim myIE As SHDocVw.InternetExplorer
Const myPageTitle As String = "SmartChoice"
Dim SmC_System As String
Dim myPageURL As String

'chose a SmC system
SmC_System = "Prod"
If SmC_System = "Prod" Then
    myPageURL = _

```

```
"http://opxscp.eame.syngenta.org/PROD/OPX2/frgolappschp01.eame.syngenta.org:8401/HOME?
dispatched=http://opxscp.eame.syngenta.org/PROD/OPX2/15.141.4.50:8100/"
Else
```

```
    myPageURL = _
    "http://opxscs.eame.syngenta.org/STAGE/OPX2/frgolappschs02.eame.syngenta.org:8404/HOME?
dispatched=http://opxscs.eame.syngenta.org/STAGE/OPX2/15.141.4.48:8100/"
End If
```

```
'IE object instantiation
Set myIE = IE_Preparation(myPageTitle, myPageURL)
Application.Wait (Now + TimeValue("0:00:01"))
```

```
'get the IE handle
Dim My_IE_hWnd As Long
My_IE_hWnd = myIE.hwnd
```

```
'show or restore IE depending on its current state
If IsIconic(My_IE_hWnd) Then
    Call ShowWindow(My_IE_hWnd, SW_RESTORE)
Else
    Call ShowWindow(My_IE_hWnd, SW_SHOW)
End If
```

```
'bring SmC IE to the foreground
SetForegroundWindow My_IE_hWnd
Application.Wait (Now + TimeValue("0:00:01"))
```

```
'IE 7 make sure you have only one tab open!!!
'or find a solution to toggle between the tabs (send key Ctrl+Tab) until you found the SmC
```

```
'wait for SmC to start up
Application.Wait (Now + TimeValue("0:00:15"))
```

```
'communicate with SmC
'-----
```

```
Dim MyActuelPos As POINTAPI
Dim myTargetPos As POINTAPI
Dim Target_X, Target_Y As Long
Dim Actuel_X, Actuel_Y As Long
```

```
'Open the module button
Target_X0 = 21 '(calibrate here)
Target_Y0 = 138 '(calibrate here)
SetCursorPos Target_X0, Target_Y0
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:01"))
```

```
'Choose the module project
'new position
Target_X = Target_X0
Target_Y = Target_Y0 + 57
```

```

SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:15"))

'loop over all virtual portfolios
Dim VirtualPortfolio_DropdownPos As Variant
VirtualPortfolio_DropdownPos = Array(79, 95, 111, 127, 143, 159)

For VirtualPortfolio_DropdownPos_Iter = LBound(VirtualPortfolio_DropdownPos) To _
    UBound(VirtualPortfolio_DropdownPos)

    'show or restore IE depending
    If IsIconic(My_IE_hWnd) Then
        Call ShowWindow(My_IE_hWnd, SW_RESTORE)
    Else
        Call ShowWindow(My_IE_hWnd, SW_SHOW)
    End If

    'bring SmC IE to the foreground
    Call SetForegroundWindow(My_IE_hWnd)

    'Open the virtual portfolio dropdown
    'new position
    Target_X = Target_X0 + 550
    Target_Y = Target_Y0 + 62
    SetCursorPos Target_X, Target_Y
    Application.Wait (Now + TimeValue("0:00:01"))
    'send mouse click
    SendMouseClicked (MOUSE_LEFT)
    Application.Wait (Now + TimeValue("0:00:01"))

    'Choose a virtual portfolio
    'new position
    Target_X = Target_X0 + 420
    Target_Y = Target_Y0 + VirtualPortfolio_DropdownPos(VirtualPortfolio_DropdownPos_Iter)
    SetCursorPos Target_X, Target_Y
    Application.Wait (Now + TimeValue("0:00:01"))
    'send mouse click
    SendMouseClicked (MOUSE_LEFT)
    Application.Wait (Now + TimeValue("0:00:10"))

    'Click the select all button
    'new position
    Target_X = Target_X0
    Target_Y = Target_Y0 + 150
    SetCursorPos Target_X, Target_Y
    Application.Wait (Now + TimeValue("0:00:01"))
    'send mouse click
    SendMouseClicked (MOUSE_LEFT)
    Application.Wait (Now + TimeValue("0:00:01"))

```



```

'wait SmC to select the projects
Application.Wait (Now + TimeValue("0:00:10"))

'open the activities or not depending on the requested report
If SmC_Report = "Activity" Then
    'click open

    'click the style

    '
Else
    'do nothing
End If

'click the download into excel button
'new position
Target_X = Target_X0 + 1019
Target_Y = Target_Y0
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)

Application.Wait (Now + TimeValue("0:00:03"))

'Minimize SmC IE
Call ShowWindow(My_IE_hWnd, SW_MINIMIZE)
Application.Wait (Now + TimeValue("0:00:01"))

'minimize IE that prepares the download
strWindowTitle = "Windows Internet Explorer provided by Syngenta"
Dim My_IE_XL_hWnd As Long
My_IE_XL_hWnd = FindWindowHandle(strWindowTitle)
If My_IE_XL_hWnd <> 0 Then
    Call ShowWindow(My_IE_XL_hWnd, SW_MINIMIZE)
    Application.Wait (Now + TimeValue("0:00:03"))
End If

'make excel active
Call z_ExcelSessionWindowNormal(mywb)
DoEvents
Call z_ExcelSessionWindowMinimized(mywb)
DoEvents
Call z_ExcelSessionWindowNormal(mywb)
DoEvents
Application.Wait (Now + TimeValue("0:00:10"))

'if there pops up a msgbox click ok
Target_X = Target_X0 + 567
Target_Y = Target_Y0 + 552
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click

```

```

SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:05"))

'minimize all Excel workbooks within the Excel session
Call z_ExcelWorkbookWindowMinimizeAll(mywb)
Application.Wait (Now + TimeValue("0:00:01"))

'minimize excel
Call z_ExcelSessionWindowMinimized(mywb)
Application.Wait (Now + TimeValue("0:00:01"))

'If IE download fails, close IE
strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet Explorer
provided by Syngenta"
Dim My_IE_XL_Error_hWnd As Long
My_IE_XL_Error_hWnd = FindWindowHandle(strWindowTitle)
Call z_CloseIE2(My_IE_XL_Error_hWnd)
Application.Wait (Now + TimeValue("0:00:01"))

Next VirtualPortfolio_DropdownPos_Iter
Stop
End Sub

Sub determineMousePos()
Dim MyActuelPos As POINTAPI
Dim myTargetPos As POINTAPI
Dim Target_X, Target_Y As Long
Dim Actuel_X, Actuel_Y As Long
'aktuelle pos abfragen
'If necessary change to VBA with ALT-Tab and start with F5 or F8
Do While 1
GetCursorPos MyActuelPos
Actuel_X = MyActuelPos.x
Actuel_Y = MyActuelPos.y
Debug.Print Actuel_X & " " & Actuel_Y
Stop
Loop
End Sub

Function Key_Alt_O()
'http://www.unet.univie.ac.at/~a7425519/programme/hex2dez.htm
Const VK_MENU = 18 '0x12
Const VK_O = 79 '0x4F
'WinKey down
keybd_event VK_MENU, 0, 0, 0
'F key down
keybd_event VK_O, 0, 0, 0
'M key up
keybd_event VK_O, 0, KEYEVENTF_KEYUP, 0
'WinKey up
keybd_event VK_MENU, 0, KEYEVENTF_KEYUP, 0
End Function

```

'-----Mouse

```
Private Declare Function SetCursorPos Lib "user32.dll" ( _  
    ByVal x As Long, _  
    ByVal y As Long) As Long
```

```
Private Declare Sub Sleep Lib "kernel32.dll" ( _  
    ByVal dwMilliseconds As Long)
```

```
Private Declare Function GetCursorPos Lib "user32.dll" ( _  
    ByRef lpPoint As POINTAPI) As Long
```

Private Type POINTAPI

    x As Long

    y As Long

End Type

```
Private Declare Sub mouse_event Lib "user32" _  
    (ByVal dwFlags As Long, ByVal dx As Long, _  
    ByVal dy As Long, ByVal cButtons As Long, _  
    ByVal dwExtraInfo As Long)
```

Public Const MOUSE\_LEFT = 0

Public Const MOUSE\_MIDDLE = 1

Public Const MOUSE\_RIGHT = 2

'-----keyboard

```
Private Declare Sub keybd_event Lib "user32.dll" (ByVal bVk As Byte, ByVal bScan As Byte, ByVal  
dwFlags As Long, _  
ByVal dwExtraInfo As Long)
```

Const VK\_STARTKEY = &H5B

Const VK\_M = 77

Const KEYEVENTF\_KEYUP = &H2

'-----Window

```
Private Declare Function PostMessage Lib "user32" _  
    Alias "PostMessageA" _  
    (ByVal hwnd As Long, _  
    ByVal wMsg As Long, _  
    ByVal wParam As Long, _  
    ByVal lParam As Long) As Long
```

Private Const WM\_CLOSE = &H10

```
Declare Function MoveWindow Lib "user32.dll" ( _  
    ByVal hwnd As Long, _  
    ByVal x As Long, _  
    ByVal y As Long, _  
    ByVal nWidth As Long, _  
    ByVal nHeight As Long, _
```

ByVal bRepaint As Long) As Long

```
Private Declare Function FindWindow Lib "user32" _  
    Alias "FindWindowA" ( _  
        ByVal IpClassName As String, _  
        ByVal IpWindowName As String) As Long
```

```
' Module Name: ModFindWindowLike  
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)  
' Written 02/06/2005
```

```
Private Declare Function EnumWindows Lib "user32" _  
    (ByVal IpEnumFunc As Long, _  
     ByVal IParam As Long) As Long
```

```
Private Declare Function GetWindowText Lib "user32" _  
    Alias "GetWindowTextA" _  
    (ByVal hwnd As Long, _  
     ByVal IpString As String, _  
     ByVal cch As Long) As Long
```

```
' Custom structure for passing in the parameters in/out of the hook enumeration function  
' Could use global variables instead, but this is nicer.
```

```
Private Type FindWindowParameters
```

```
    strTitle As String 'INPUT  
    hwnd As Long      'OUTPUT
```

```
End Type
```

```
' Module Name: Modz_SetForegroundWindow  
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)  
' Written 02/06/2005
```

```
Private Declare Function SetForegroundWindow Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Private Declare Function GetWindowThreadProcessId Lib "user32" _  
    (ByVal hwnd As Long, _  
     lpdwProcessId As Long) As Long
```

```
Private Declare Function IsIconic Lib "user32" _  
    (ByVal hwnd As Long) As Long
```

```
Private Declare Function ShowWindow Lib "user32" _  
    (ByVal hwnd As Long, _  
     ByVal nCmdShow As Long) As Long
```

```
Private Declare Function AttachThreadInput Lib "user32" _  
    (ByVal idAttach As Long, _  
     ByVal idAttachTo As Long, _  
     ByVal fAttach As Long) As Long
```

```
Private Declare Function GetForegroundWindow Lib "user32" _  
    () As Long
```

```
Private Const SW_RESTORE = 9
```

```
Private Const SW_SHOW = 5
```

```
' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel
```

```
,
```

```
' sTitel: muss nicht der exakte Fenstertitel sein
```

```
'     hier kann bspw. auch nur der Anfang des Fenstertitel
```

```
'     angegeben werden, z.B.: Fenstertitel*
```

```
,
```

```
' benötigte API-Deklarationen
```

```
Private Declare Function GetWindowTextLength Lib "user32" _
```

```
    Alias "GetWindowTextLengthA" ( _
```

```
    ByVal hwnd As Long) As Long
```

```
Private Declare Function GetWindow Lib "user32" ( _
```

```
    ByVal hwnd As Long, _
```

```
    ByVal wCmd As Long) As Long
```

```
Private Const GW_HWNDNEXT = 2
```

```
Declare Function FindWindowEx Lib "user32.dll" _
```

```
    Alias "FindWindowExA" ( _
```

```
    ByVal hwndParent As Long, _
```

```
    ByVal hwndChildAfter As Long, _
```

```
    ByVal lpszClass As String, _
```

```
    ByVal lpszWindow As String) As Long
```

```
'-----Mouse
```

```
'Die nachfolgende Prozedur simuliert den gewünschten Mausklick.
```

```
Public Sub z_SendMouseClicked(ByVal mButton As Long)
```

```
    Const MOUSEEVENTF_LEFTDOWN = &H2
```

```
    Const MOUSEEVENTF_LEFTUP = &H4
```

```
    Const MOUSEEVENTF_MIDDLEDOWN = &H20
```

```
    Const MOUSEEVENTF_MIDDLEUP = &H40
```

```
    Const MOUSEEVENTF_RIGHTDOWN = &H8
```

```
    Const MOUSEEVENTF_RIGHTUP = &H10
```

```
    If (mButton = MOUSE_LEFT) Then
```

```
        Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
```

```
        Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
```

```
    ElseIf (mButton = MOUSE_MIDDLE) Then
```

```
        Call mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
```

```
        Call mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
```

```
    Else
```

```
        Call mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
```

```

    Call mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
End If
End Sub

```

```

Sub z_SendMausDoubleClick(ByVal mButton As Long)
z_SendMouseClicked (mButton)
z_SendMouseClicked (mButton)
End Sub

```

```

Function z_SetMousePosAndLeftClick(Target_X0 As Long, Target_Y0 As Long, _
    Target_DeltaX As Long, Target_DeltaY As Long)
Target_X = Target_X0 + Target_DeltaX
Target_Y = Target_Y0 + Target_DeltaY
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
z_SendMouseClicked (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:01"))
End Function

```

'-----keyobard

```

Function z_ShowDesktop()
'Keyboard: Windows button + M button shows the desktop
'Do not test with debug F8 -> VBA Windows hides!!!
'http://msdn.microsoft.com/en-us/library/ms646304(VS.85).aspx
'http://msdn.microsoft.com/en-us/library/dd375731(v=VS.85).aspx

```

```

'WinKey down
keybd_event VK_STARTKEY, 0, 0, 0
'M key down
keybd_event VK_M, 0, 0, 0
'M key up
keybd_event VK_M, 0, KEYEVENTF_KEYUP, 0
'WinKey up
keybd_event VK_STARTKEY, 0, KEYEVENTF_KEYUP, 0

```

```

'do not minimiz form itself
'Me.WindowState = vbMaximized
'Me.WindowState = vbNormal[/b]
End Function

```

'-----Window

```

' Ermittelt das Handle eines Fensters anhand dessen Fenstertitel
'
' sTitel: muss nicht der exakte Fenstertitel sein
'       hier kann bspw. auch nur der Anfang des Fenstertitel
'       angegeben werden, z.B.: Fenstertitel*
'

```

```

Public Function z_FindWindowHandle(ByVal sTitle As String) As Long
Dim lngHWND As Long
Dim sText As String

```

```

' Handel des ersten Fensters
lngHwnd = FindWindow(vbNullString, vbNullString)
' alle Fenster durchlaufen
Do While lngHwnd <> 0

    ' Fenstertitel ermitteln
    sText = z_GetWindowTitle(lngHwnd)
    'Debug.Print lngHwnd & " " & sText
    If Len(sText) > 0 And LCase$(sText) Like LCase$(sTitle) Then
        z_FindWindowHandle = lngHwnd: Exit Do
    End If

    ' Handel des nächsten Fensters
    lngHwnd = GetWindow(lngHwnd, GW_HWNDNEXT)
Loop
End Function

' Hilfsfunktion zum Ermitteln des Fenstertitels
Public Function z_GetWindowTitle(ByVal hwnd As Long) As String
    Dim lResult As Long
    Dim sTemp As String

    lResult = GetWindowTextLength(hwnd) + 1
    sTemp = Space(lResult)
    lResult = GetWindowText(hwnd, sTemp, lResult)
    z_GetWindowTitle = left(sTemp, Len(sTemp) - 1)
End Function

' Module Name: ModFindWindowLike
' (c) 2005 Wayne Phillips (http://www.everythingaccess.com)
' Written 02/06/2005
Public Function z_FindWindowLike(strWindowTitle As String) As Long

    'We'll pass a custom structure in as the parameter to store our result...
    Dim Parameters As FindWindowParameters
    Parameters.strTitle = strWindowTitle ' Input parameter

    Call EnumWindows(AddressOf EnumWindowProc, VarPtr(Parameters))

    z_FindWindowLike = Parameters.hwnd

End Function

Private Function EnumWindowProc(ByVal hwnd As Long, _
    lParam As FindWindowParameters) As Long

    Dim strWindowTitle As String

    strWindowTitle = Space(260)
    Call GetWindowText(hwnd, strWindowTitle, 260)
    strWindowTitle = TrimNull(strWindowTitle) ' Remove extra null terminator

```

If strWindowTitle Like IParam.strTitle Then

    IParam.hwnd = hwnd 'Store the result for later.

    EnumWindowProc = 0 'This will stop enumerating more windows

Else

    EnumWindowProc = 1

End If

End Function

' Module Name: ModSetForegroundWindow

' (c) 2005 Wayne Phillips (<http://www.everythingaccess.com>)

' Written 02/06/2005

Private Function TrimNull(strNullTerminatedString As String)

    Dim lngPos As Long

    'Remove unnecessary null terminator

    lngPos = InStr(strNullTerminatedString, Chr\$(0))

    If lngPos Then

        TrimNull = left\$(strNullTerminatedString, lngPos - 1)

    Else

        TrimNull = strNullTerminatedString

    End If

End Function

Public Function z\_SetForegroundWindow(strWindowTitle As String) As Boolean

    Dim MyAppHWND As Long

    Dim CurrentForegroundThreadID As Long

    Dim NewForegroundThreadID As Long

    Dim lngRetVal As Long

    Dim blnSuccessful As Boolean

    MyAppHWND = z\_FindWindowLike(strWindowTitle)

    If MyAppHWND <> 0 Then

        'We've found the application window by the caption

        CurrentForegroundThreadID = GetWindowThreadProcessId(GetForegroundWindow(), ByVal 0&)

        NewForegroundThreadID = GetWindowThreadProcessId(MyAppHWND, ByVal 0&)

        'AttachThreadInput is used to ensure SetForegroundWindow will work

        'even if our application isn't currently the foreground window

        '(e.g. an automated app running in the background)

        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, True)

        lngRetVal = SetForegroundWindow(MyAppHWND)

        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, False)

    If lngRetVal <> 0 Then

        'Now that the window is active, let's restore it from the taskbar

        If IsIconic(MyAppHWND) Then



```

        Call ShowWindow(MyAppHWnd, SW_RESTORE)
    Else
        Call ShowWindow(MyAppHWnd, SW_SHOW)
    End If
    blnSuccessful = True
Else
    MsgBox "Found the window, but failed to bring it to the foreground!"
End If
Else
    'Failed to find the window caption
    'Therefore the app is probably closed.
    MsgBox "Application Window '" + strWindowTitle + "' not found!"
End If
z_SetForegroundWindow = blnSuccessful
End Function
Public Function z_SetForegroundWindow2(MyAppHWnd As Long) As Boolean
    Dim CurrentForegroundThreadID As Long
    Dim NewForegroundThreadID As Long
    Dim lngRetVal As Long
    Dim blnSuccessful As Boolean
    If MyAppHWnd <> 0 Then
        'We've found the application window by the caption
        CurrentForegroundThreadID = GetWindowThreadProcessId(GetForegroundWindow(), ByVal 0&)
        NewForegroundThreadID = GetWindowThreadProcessId(MyAppHWnd, ByVal 0&)
        'AttachThreadInput is used to ensure SetForegroundWindow will work
        'even if our application isn't currently the foreground window
        '(e.g. an automated app running in the background)
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, True)
        lngRetVal = SetForegroundWindow(MyAppHWnd)
        Call AttachThreadInput(CurrentForegroundThreadID, NewForegroundThreadID, False)
        If lngRetVal <> 0 Then
            'Now that the window is active, let's restore it from the taskbar
            If IsIconic(MyAppHWnd) Then
                Call ShowWindow(MyAppHWnd, SW_RESTORE)
            Else
                Call ShowWindow(MyAppHWnd, SW_SHOW)
            End If
            blnSuccessful = True
        Else
            MsgBox "Found the window, but failed to bring it to the foreground!"
        End If
    Else
        'Failed to find the window caption
        'Therefore the app is probably closed.
        MsgBox "Application Window '" + strWindowTitle + "' not found!"
    End If
    z_SetForegroundWindow2 = blnSuccessful
End Function

```

```

'-----Microsoft Internet Controls (Tools>References: Microsoft Internet Controls)
'returns new instance of Internet Explorer
Function GetNewIE() As SHDocVw.InternetExplorer

```

```

'create new IE instance
Set GetNewIE = New SHDocVw.InternetExplorer
'start with a blank page
GetNewIE.Navigate2 "about:Blank"
End Function

```

```

'loads a web page and returns True or False depending on
'whether the page could be loaded or not
Function LoadWebPage(i_IE As SHDocVw.InternetExplorer, _
    i_URL As String) As Boolean
With i_IE
'open page
.Navigate i_URL
'Window security
If i_URL = "smartchoice.stg.intra" Then
'here code that fills the pop up
End If
'wait until IE finished loading the page
Do While .ReadyState <> READYSTATE_COMPLETE
Application.Wait Now + TimeValue("0:00:01")
Loop
'check if page could be loaded
'(does not work out if the page is relocated)
'this is a workaround. Better: do check wheter a window "SmartChoice" exists!!
If i_URL = "smartchoice.pro.intra" Or i_URL = "smartchoice.stg.intra" Then
LoadWebPage = True
Else
If .Document.URL = i_URL Then
LoadWebPage = True
End If
End If
End With
End Function

```

```

Function z_IE_WaitUntilReadyStateComplete(i_IE As SHDocVw.InternetExplorer) As Integer
Dim sec As Integer
sec = 0
'wait until IE finished loading the page
Do While i_IE.ReadyState <> READYSTATE_COMPLETE
Application.Wait Now + TimeValue("0:00:01")
sec = sec + 1
Loop
End Function

```

```

Function z_IE_WaitUntilNewWindowExists(strWindowTitle As String, secMax As Integer) As Integer
Dim sec As Integer
sec = 0
'Do While MyAppHWnd = 0
' MyAppHWnd = z_FindWindowLike(strWindowTitle)
' Application.Wait Now + TimeValue("0:00:01")
' sec = sec + 1
' If sec = secMax Then
' Exit Function

```

```

' End If
'Loop
Do While MyAppHWnd = 0
    MyAppHWnd = z_FindWindowHandle(strWindowTitle)
    Application.Wait Now + TimeValue("0:00:01")
    sec = sec + 1
    If sec = secMax Then
        z_IE_WaitUntilNewWindowExists = secMax
        Exit Function
    End If
Loop
z_IE_WaitUntilNewWindowExists = sec
End Function

'finds an open IE site by checking the URL
Function GetOpenIEByURL(ByVal i_URL As String) _
    As SHDocVw.InternetExplorer

Dim objShellWindows As New SHDocVw.ShellWindows

'ignore errors when accessing the document property
On Error Resume Next
'loop over all Shell-Windows
For Each GetOpenIEByURL In objShellWindows
    'if the document is of type HTMLDocument, it is an IE window
    If TypeName(GetOpenIEByURL.Document) = "HTMLDocument" Then
        'check the URL
        If GetOpenIEByURL.Document.URL = i_URL Then
            'leave, we found the right window
            Exit Function
        End If
    End If
Next
End Function

'finds an open IE site by checking the title
Function GetOpenIEByTitle(i_Title As String, _
    Optional ByVal i_ExactMatch As Boolean = True) _
    As SHDocVw.InternetExplorer

Dim objShellWindows As New SHDocVw.ShellWindows

If i_ExactMatch = False Then i_Title = "*" & i_Title & "*"
'ignore errors when accessing the document property
On Error Resume Next
'loop over all Shell-Windows
For Each GetOpenIEByTitle In objShellWindows
    'if the document is of type HTMLDocument, it is an IE window
    If TypeName(GetOpenIEByTitle.Document) = "HTMLDocument" Then
        'check the title
        If GetOpenIEByTitle.Document.Title Like i_Title Then
            'leave, we found the right window

```

```
Exit Function
End If
End If
Next
End Function
```

```
Function IE_Preparation(myPageTitle As String, myPageURL As String) _
    As SHDocVw.InternetExplorer
```

```
'IE object instantiation
Dim myIE As SHDocVw.InternetExplorer

'check if page is already open
Set myIE = GetOpenIEByTitle(myPageTitle, False)
```

```
'check if page is already open
'(does not work if the page is relocated)
'Set myIE = GetOpenIEByURL(myPageURL)
```

```
'if the page is not open then try to open it
If myIE Is Nothing Then
    'page isn't open yet
    'create new IE instance
    Set myIE = GetNewIE
    'make IE window visible
    myIE.Visible = True
    'IE move and resize
    Call IE_MoveAndResize(myIE, 0, 0, 1200, 900)
    Application.Wait (Now + TimeValue("0:00:05"))
    'load page
    If LoadWebPage(myIE, myPageURL) = False Then
        'page wasn't loaded
        MsgBox "Couldn't open page"
        Exit Function
    End If
End If
```

```
'move and resize the window (do it before you start the URL)
'Dim nhWnd As Long
'nhWnd1 = FindWindow(vbNullString, "SmartChoice - Windows Internet Explorer provided by Syngenta")
'nhWnd1 = myIE.hWnd
'If nhWnd1 <> 0 Then
'    MoveWindow nhWnd1, 0, 0, 1200, 900, 1
'End If
```

```
Set IE_Preparation = myIE
End Function
```

```
Function IE_MoveAndResize(ByRef IE As SHDocVw.InternetExplorer, _
    top As Variant, left As Variant, _
    width As Variant, height As Variant)
'move and resize the window
```

```

Dim nhWnd As Long
nhWnd = IE.hwnd
If nhWnd <> 0 Then
    MoveWindow nhWnd, top, left, width, height, 1
End If

```

```

End Function

```

```

Function z_CloseIE(strWindowTitle As String)
    Const WM_CLOSE = &H10
    Dim hwnd As Long
    hwnd = z_FindWindowHandle(strWindowTitle)
    If hwnd <> 0 Then
        'bring to the foreground
        SetForegroundWindow hwnd
        'close the IE window
        PostMessage hwnd, WM_CLOSE, 0&, 0&
    End If
End Function

```

```

Function z_CloseIE2(hwnd As Long)
    Const WM_CLOSE = &H10
    If hwnd <> 0 Then
        'bring to the foreground
        SetForegroundWindow hwnd
        'close the IE window
        PostMessage hwnd, WM_CLOSE, 0&, 0&
    End If
End Function

```

```

Public Function z_OpenAndActivateWb(wbname As String, wbpas As String, ByRef Wb As
Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Objective: Open the Workbook RD.xlsb if not already open and activate it.
'look if wbName is existent in workbooks list
Dim i As Long
For i = Workbooks.Count To 1 Step -1
    If Workbooks(i).Name = wbname Then Exit For
Next
'if wbName is existent in workbooks list, then i<>0-> activate workbook
'if wbName is not existent then i=0-> open workbook, activate workbook
If i <> 0 Then
    Set Wb = GetObject(wbpas & wbname)
    Wb.Activate
Else
    Set Wb = Workbooks.Open(wbpas & wbname)
    Wb.Activate
End If
End Function

```

```
Function z_WorkbookNewOrOpenOrActivate(wbname As String, wbpas As String, ByRef Wb As Workbook)
```

```
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

```
'Date: 19.10.2011
```

```
'Objective: Open the Workbook RD.xlsb if not already open and activate it.
```

```
'look if wbName is existent in workbooks list
```

```
Dim i As Long
```

```
For i = Workbooks.Count To 1 Step -1
```

```
    If Workbooks(i).Name = wbname Then Exit For
```

```
Next
```

```
'if wbName is existent in workbooks list, then i<>0-> activate workbook
```

```
'if wbName is not existent then i=0-> open workbook, activate workbook
```

```
If i <> 0 Then
```

```
    Set Wb = GetObject(wbpas & wbname)
```

```
    Wb.Activate
```

```
Else
```

```
    On Error GoTo NewWB
```

```
    Set Wb = Workbooks.Open(wbpas & wbname)
```

```
    Wb.Activate
```

```
    On Error GoTo 0
```

```
End If
```

```
Exit Function
```

```
NewWB:
```

```
Set Wb = Workbooks.Add
```

```
Dim FileFormatValue As Integer
```

```
If wbname <> Empty Then
```

```
    Select Case LCase(Right(wbname, Len(wbname) - InStrRev(wbname, ".", , 1)))
```

```
        Case "xls": FileFormatValue = 56
```

```
        Case "xlsx": FileFormatValue = 51
```

```
        Case "xlsm": FileFormatValue = 52
```

```
        Case "xlsb": FileFormatValue = 50
```

```
        Case Else: FileFormatValue = 0
```

```
    End Select
```

```
End If
```

```
Wb.SaveAs filename:=wbpas & wbname, FileFormat:=FileFormatValue
```

```
End Function
```

```
Function z_ExcelSessionWindowMinimized(Optional ByRef Wb As Workbook)
```

```
'Fenster der Excel session
```

```
If Wb Is Nothing Then
```

```
    Application.WindowState = xlMinimized
```

```
Else
```

```
    Wb.Application.WindowState = xlMinimized
```

```
End If
```

```
End Function
```

```
Function z_ExcelSessionWindowNormal(Optional ByRef Wb As Workbook)
```

```
'Fenster der Excel session
```

```
If Wb Is Nothing Then
```

```
    Application.WindowState = xlNormal
```

```
Else
```

```
    Wb.Application.WindowState = xlNormal
```

```
    Wb.Activate
```

```
End If
```

End Function

```
Sub z_ExcelSessionWindowMoveAndResize(Optional ByRef Wb As Workbook, _  
    Optional top As Variant, Optional left As Variant, _  
    Optional width As Variant, Optional height As Variant)
```

```
If Wb Is Nothing Then  
    If top <> Empty Then  
        Application.top = top  
    End If  
    If left <> Empty Then  
        Application.left = left  
    End If  
    If width <> Empty Then  
        Application.width = width  
    End If  
    If height <> Empty Then  
        Application.height = height  
    End If
```

```
Else  
    If top <> Empty Then  
        Wb.Application.top = CInt(top)  
    End If  
    If left <> Empty Then  
        Wb.Application.left = CInt(left)  
    End If  
    If width <> Empty Then  
        Wb.Application.width = CInt(width)  
    End If  
    If height <> Empty Then  
        Wb.Application.height = CInt(height)  
    End If  
End If
```

End Sub

```
Function z_ExcelWorkbookWindowMinimized(ByRef Wb As Workbook)
```

```
'Fenster innerhalb der Excel session (workbooks)  
If Windows(Wb.Name).Visible Then  
    ActiveWindow.WindowState = xlMinimized  
End If
```

End Function

```
Function z_ExcelWorkbookWindowNormal(ByRef Wb As Workbook)
```

```
'Fenster innerhalb der Excel session (workbooks)  
If Windows(Wb.Name).Visible Then  
    ActiveWindow.WindowState = xlMaximized  
End If
```

End Function

```
Function z_ExcelWorkbookWindowMinimizeAll(ByRef Wb_ref As Workbook)
```

```
Application.ScreenUpdating = False  
Dim Wb As Workbook  
For Each Wb In Wb_ref.Application.Workbooks ' Workbooks  
    'only those with status visible  
    If Windows(Wb.Name).Visible Then  
        ActiveWindow.WindowState = xlMinimized
```

```

End If
Next
Application.ScreenUpdating = True
End Function

```

'SmC Prozesse werden jeden Morgen um 2 Uhr gestoppt und bis 4 Uhr neu gestartet.  
 'Am besten dieses Programm um 5 Uhr morgens starten, dann sind die Leitungen frei,  
 'der Cache leer usw.

```

Sub IE_CreateReport()
'close all windows/ Show the desktop
Call z_ShowDesktop
Application.Wait (Now + TimeValue("0:00:01"))

```

```

'choose a mode
'-----
Dim test As Integer
test = 0 ' "No=0, Yes=1"
'-----

```

```

'choose Smc user rights with and without read only
'-----
Dim delta1 As Integer
Dim delta2 As Integer
delta1 = 0 'read only: 0, all rights: 17
delta2 = 0 'read only: 0, all rights: 56
'-----

```

```

'choose a report Pls or Activitys
'-----
Dim SmC_Report As Integer
SmC_Report = 1 ' "Pls=0, Activity=1"
'-----

```

```

'add, open or activate an Excel workbook (and session)
Dim mywb As Workbook
Dim wbpas As String
Dim wbnas As String
wbpas = "C:\Users\t740698\Desktop\"
wbnas = "SmC_Download" & "_" & VBA.DateTime.Day(Now()) & "_" & _
      VBA.DateTime.Month(Now()) & "_" & VBA.DateTime.Year(Now()) & ".xlsb"
'wbnas = "SmC_Download.xlsb"
Call z_WorkbookNewOrOpenOrActivate(wbnas, wbpas, mywb)
Application.Wait (Now + TimeValue("0:00:03"))

```

```

'move and resize excel session
Call z_ExcelSessionWindowNormal(mywb)
Call z_ExcelSessionWindowMoveAndResize(mywb, "0", "0", "700", "600")
Application.Wait (Now + TimeValue("0:00:03"))

```

```

'minimize all Excel workbooks within the Excel session except mywb
Call z_ExcelWorkbookWindowMinimizeAll(mywb)
Call z_ExcelWorkbookWindowNormal(mywb)

```



```

'prepare the SmC window in an IE object
'IE object declaration
Dim myIE As SHDocVw.InternetExplorer
Const myPageTitle As String = "SmartChoice"
Dim SmC_System As Integer
Dim myPageURL As String

'Name of the download window
Dim strWindowTitle As String

'chose a SmC system
'-----
SmC_System = 1 'Prod=1, Stage=0
'-----
If SmC_System = 1 Then
    myPageURL = _
        "smartchoice.pro.intra"
Else
    myPageURL = _
        "smartchoice.stg.intra"
End If

'IE object instantiation
Set myIE = IE_Preparation(myPageTitle, myPageURL)
Application.Wait (Now + TimeValue("0:00:01"))

'get the IE handle
Dim My_IE_hWnd As Long
My_IE_hWnd = myIE.hwnd

'show or restore IE depending on its current state
If IsIconic(My_IE_hWnd) Then
    Call ShowWindow(My_IE_hWnd, SW_RESTORE)
Else
    Call ShowWindow(My_IE_hWnd, SW_SHOW)
End If

'bring SmC IE to the foreground
SetForegroundWindow My_IE_hWnd
Application.Wait (Now + TimeValue("0:00:01"))

'IE 7 make sure you have only one tab open!!!
'or find a solution to toggle between the tabs (send key Ctrl+Tab) until you found the SmC

'wait for SmC to start up
If test Then
    Application.Wait (Now + TimeValue("0:00:01"))
Else
    Application.Wait (Now + TimeValue("0:00:15"))
End If

'communicate with SmC

```

```

'-----
Dim Target_X0 As Long
Dim Target_Y0 As Long

'Open the module button
Target_X0 = 21 '(calibrate here)
Target_Y0 = 137 '(calibrate here)
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
Application.Wait (Now + TimeValue("0:00:01"))

'Choose the module "project"
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
Application.Wait (Now + TimeValue("0:00:10"))

'Choose the "BySyngentaPortfolio" button
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 144, 123)
If test Then
    Application.Wait (Now + TimeValue("0:00:01"))
Else
    Application.Wait (Now + TimeValue("0:00:10"))
End If

'Choose the "VirtualPortfolio" button
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 238 + delta1, 63)
Application.Wait (Now + TimeValue("0:00:01"))

'Choose the "Reset" button
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 238 + delta1, 148)
If test Then
    Application.Wait (Now + TimeValue("0:00:01"))
Else
    Application.Wait (Now + TimeValue("0:00:10"))
End If

'loop over all virtual portfolios
Dim VirtualPortfolio_DropdownSecondPos As Long
VirtualPortfolio_DropdownSecondPos = 97
'-----
Const TotalVirtualPortfolios = 16 'from 0 to 16
'-----
Dim Array_FailedDownloads(0 To TotalVirtualPortfolios) As Variant
Dim FailedDownloads_iter As Integer
FailedDownloads_iter = 1

For VirtualPortfolio_DropdownPos_Iter = 0 To TotalVirtualPortfolios

    If 1 Then ' VirtualPortfolio_DropdownPos_Iter <> 0 Then
        'show or restore IE depending on its current state

```

```

If IsIconic(My_IE_hWnd) Then
    Call ShowWindow(My_IE_hWnd, SW_RESTORE)
Else
    Call ShowWindow(My_IE_hWnd, SW_SHOW)
End If
'bring SmC IE to the foreground
Call SetForegroundWindow(My_IE_hWnd)
End If

```

```

'To be sure SmC is in an idle state before going back to the portfolio module
Application.Wait (Now + TimeValue("0:00:10"))

```

```

'close the activities or not depending on the requested report
'bring SmC IE back to the modul "project"
If VirtualPortfolio_DropdownPos_Iter <> 0 Then
    If SmC_Report = 1 Then
        'Open the module button
        'new position and mouse click
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 0)
        Application.Wait (Now + TimeValue("0:00:01"))
        'Choose the module "project"
        'new position and mouse click
        Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 57)
        Application.Wait (Now + TimeValue("0:00:10"))
    End If
End If

```

```

'Open the virtual portfolio dropdown
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 533 + delta1, 63)
Application.Wait (Now + TimeValue("0:00:01"))

```

```

'Choose a virtual portfolio
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 338 + delta1,
VirtualPortfolio_DropdownSecondPos)
Application.Wait (Now + TimeValue("0:00:10"))

```

```

'Click the "select all" button
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 0, 142)
Application.Wait (Now + TimeValue("0:00:10"))

```

```

'open the activities or not depending on the requested report
If SmC_Report = 1 Then
    'click the "open" button
    'new position and mouse click
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 120 + delta2, 64)
    Application.Wait (Now + TimeValue("0:00:01"))
    'click the "open selected projects" button
    'new position and mouse click
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 131 + delta2, 91)
    Application.Wait (Now + TimeValue("0:00:15"))

```

```

'not used if the "R&D Reporting Master Data Set" style is set as the default style
If 0 Then
    'click the "style" button
    'new position and mouse click
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 676, 135)
    Application.Wait (Now + TimeValue("0:00:01"))
    'choose the "R&D Reporting Master Data Set" style
    'new position and mouse click
    Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 580, 251)
    Application.Wait (Now + TimeValue("0:00:10"))
End If
Else
    'do nothing
End If

'click the "download into Excel" button
'new position and mouse click
Dim sec As Integer
Dim secMax As Integer
'-----
secMax = 35
'-----

Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 1019, 0)
If SmC_Report = 1 Then
    strWindowTitle = "Windows Internet Explorer provided by Syngenta"
    sec = z_IE_WaitUntilNewWindowExists(strWindowTitle, secMax)
    Debug.Print VirtualPortfolio_DropdownPos_Iter + 1 & " " & sec
    mywb.Worksheets("Sheet1").Cells(VirtualPortfolio_DropdownPos_Iter + 2, 5) =
VirtualPortfolio_DropdownPos_Iter + 1
    mywb.Worksheets("Sheet1").Cells(VirtualPortfolio_DropdownPos_Iter + 2, 6) = sec
    Application.Wait (Now + TimeValue("0:00:03"))
    'If IE download fails, close IE and wait
    'to be sure SmC is in an idle state before going back to the portfolio module
    If sec = secMax Then
        'wait until SmC is in an idle state
        Application.Wait (Now + TimeValue("0:03:01")) 'ev. enhance to 10 minutes
        strWindowTitle = "HTTP 500 Internal Server Error - Windows Internet Explorer provided by
Syngenta"
        Dim My_IE_XL_HTTP500_hWnd As Long
        My_IE_XL_HTTP500_hWnd = z_FindWindowHandle(strWindowTitle)
        If My_IE_XL_HTTP500_hWnd <> 0 Then
            Call z_CloseIE2(My_IE_XL_HTTP500_hWnd)
            Application.Wait (Now + TimeValue("0:00:01"))
            Array_FailedDownloads(FailedDownloads_iter) = VirtualPortfolio_DropdownPos_Iter + 1
            Debug.Print "HTTP500 " & FailedDownloads_iter + 1 & " " &
VirtualPortfolio_DropdownPos_Iter + 1
            mywb.Worksheets("Sheet1").Cells(FailedDownloads_iter + 1, 1) = FailedDownloads_iter
            mywb.Worksheets("Sheet1").Cells(FailedDownloads_iter + 1, 2) =
VirtualPortfolio_DropdownPos_Iter + 1
            mywb.Worksheets("Sheet1").Cells(FailedDownloads_iter + 1, 3) = "HTTP500"
            FailedDownloads_iter = FailedDownloads_iter + 1
        End If
    End If
End If

```

```

Else
    'In case the time after the download window occurs is not too long (>4 seconds)
    'it seems that the pop up windows can be avoided
    Application.Wait (Now + TimeValue("0:00:01"))
End If

'Minimize SmC IE
Call ShowWindow(My_IE_hWnd, SW_MINIMIZE)
Application.Wait (Now + TimeValue("0:00:01"))

'minimize IE that prepares the download
strWindowTitle = "Windows Internet Explorer provided by Syngenta"
Dim My_IE_XL_hWnd As Long
Dim My_IE_XL_Child_hWnd As Long
Dim MyChildName As String
My_IE_XL_hWnd = z_FindWindowHandle(strWindowTitle)
If My_IE_XL_hWnd <> 0 Then
    'find child window pop ups
    My_IE_XL_Child_hWnd = FindPopUpWindow(strWindowTitle)
    If My_IE_XL_Child_hWnd <> 0 Then
        'do the clicks on the child window or where it is
        MyChildName = z_GetWindowTitle(My_IE_XL_Child_hWnd)
        Debug.Print MyChildName
        'Stop
        If MyChildName = "" Then
            'Make clicks depending on the name
        Else
            'Make clicks depending on the name
        End If
    End If
    'minimize IE
    Call ShowWindow(My_IE_XL_hWnd, SW_MINIMIZE)
    DoEvents
    Application.Wait (Now + TimeValue("0:00:03"))
End If

'if there pops up a msgbox click ok
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 567, 552)
Application.Wait (Now + TimeValue("0:00:01"))

'make excel active
'(In case no error occurred during the download,
'these steps completes the download and closes the IE that prepares the download)
Call z_ExcelSessionWindowNormal(mywb)
Call z_ExcelWorkbookWindowNormal(mywb)
Application.Wait (Now + TimeValue("0:00:01"))
mywb.Activate
DoEvents
Call z_ExcelSessionWindowMinimized(mywb)
DoEvents
Call z_ExcelSessionWindowNormal(mywb)
mywb.Activate

```

```

DoEvents
Application.Wait (Now + TimeValue("0:00:05"))

'if there pops up a msgbox click ok
'new position and mouse click
Call z_SetMousePosAndLeftClick(Target_X0, Target_Y0, 567, 552)
Application.Wait (Now + TimeValue("0:00:01"))

'minimize all Excel workbooks within the Excel session
Call z_ExcelWorkbookWindowMinimizeAll(mywb)
Application.Wait (Now + TimeValue("0:00:01"))

'If IE download fails, close IE and store the error
strWindowTitle = "Internet Explorer cannot display the webpage - Windows Internet Explorer
provided by Syngenta"
Dim My_IE_XL_Error_hWnd As Long
My_IE_XL_Error_hWnd = z_FindWindowHandle(strWindowTitle)
If My_IE_XL_Error_hWnd <> 0 Then
    Call z_CloseIE2(My_IE_XL_Error_hWnd)
    Application.Wait (Now + TimeValue("0:00:01"))
    Array_FailedDownloads(FailedDownloads_iter) = VirtualPortfolio_DropdownPos_iter + 1
    Debug.Print "CannotDisplayWebpage   " & FailedDownloads_iter + 1 & "   " &
VirtualPortfolio_DropdownPos_iter + 1
    mywb.Worksheets("Sheet1").Cells(FailedDownloads_iter + 1, 1) = FailedDownloads_iter
    mywb.Worksheets("Sheet1").Cells(FailedDownloads_iter + 1, 2) =
VirtualPortfolio_DropdownPos_iter + 1
    mywb.Worksheets("Sheet1").Cells(FailedDownloads_iter + 1, 3) = "Cannot display webpage"
    FailedDownloads_iter = FailedDownloads_iter + 1
End If
'FindPopUpWindow("Windows Internet Explorer provided by Syngenta")
'Stop
Next VirtualPortfolio_DropdownPos_iter
Stop
End Sub

Function FindPopUpWindow(optionalParentName As String, Optional hwndParent As Long) As Long
Dim WindowName As String
WindowName = vbNullString
If hwndParent = 0 Then
    hwndParent = z_FindWindowHandle(ParentName)
End If
hWndChild = FindWindowEx(hwndParent, 0&, vbNullString, WindowName)
FindPopUpWindow = hWndChild
'when found do the clicks
End Function

Sub MoveWbSheetsIntoAnotherWb()
Dim mywb As Workbook
Dim wbpath As String
Dim wdbname As String
wbpath = "C:\Users\t740698\Desktop\"
wdbname = "SmC_Download" & "_" & VBA.DateTimes.Day(Now()) & "_" & _
    VBA.DateTimes.Month(Now()) & "_" & VBA.DateTimes.Year(Now()) & ".xlsb"

```

```

'open or activate mywb
Call z_WorkbookNewOrOpenOrActivate(wbname, wbpPath, mywb)
'bring the Excel session into xlNormal
z_ExcelSessionWindowNormal (mywb)
'rename logfile
mywb.Sheets("Sheet1").Select
mywb.Sheets("Sheet1").Name = "Logfile"
'delete empty sheets
Call z_DeleteWbSheet(mywb, "Sheet2")
Call z_DeleteWbSheet(mywb, "Sheet3")
'move all windows into mywb
z_MoveWbSheetsIntoAnotherWb (mywb)
'save workbook
mywb.Save
End Sub

```

```

Function z_MoveWbSheetsIntoAnotherWb(Wb_ref As Workbook)
    Dim Wb As Workbook
    For Each Wb In Wb_ref.Application.Workbooks ' Workbooks
        'move all Windows to Wb_ref
        Windows(Wb.Name).Activate
        ActiveWindow.WindowState = xlNormal
        ActiveWindow.WindowState = xlNormal
        Sheets(Wb.Name).Select
        Sheets(Wb.Name).Move After:=Wb_ref.Sheets(Sheets.Count)
    Next
End Function

```

```

Function z_DeleteWbSheet(Wb As Workbook, Sh As String)
    Sheets(Sh).Select
    ActiveWindow.SelectedSheets.Delete
End Function

```

```

Sub determineMousePos()
    Dim MyActuelPos As POINTAPI
    Dim myTargetPos As POINTAPI
    Dim Target_X, Target_Y As Long
    Dim Actuel_X, Actuel_Y As Long
    'aktuelle pos abfragen
    'If necessary change to VBA with ALT-Tab and start with F5 or F8
    Do While 1
        GetCursorPos MyActuelPos
        Actuel_X = MyActuelPos.x
        Actuel_Y = MyActuelPos.y
        Debug.Print Actuel_X & " " & Actuel_Y
    Stop
    Loop
End Sub

```

```

Function Key_Alt_O()
'http://www.unet.univie.ac.at/~a7425519/programme/hex2dez.htm
Const VK_MENU = 18 '0x12

```

```

Const VK_O = 79 '0x4F
'WinKey down
keybd_event VK_MENU, 0, 0, 0
'F key down
keybd_event VK_O, 0, 0, 0
'M key up
keybd_event VK_O, 0, KEYEVENTF_KEYUP, 0
'WinKey up
keybd_event VK_MENU, 0, KEYEVENTF_KEYUP, 0
End Function

```

```

Private Declare Sub keybd_event Lib "user32.dll" (ByVal bVk As Byte, ByVal bScan As Byte, ByVal
dwFlags As Long, _
ByVal dwExtraInfo As Long)

```

```

Function z_ShowDesktop()
'Keyboard: Windows button + M button shows the desktop
'Do not test with debug F8 -> VBA Windows hides!!!
'http://msdn.microsoft.com/en-us/library/ms646304(VS.85).aspx
'http://msdn.microsoft.com/en-us/library/dd375731(v=VS.85).aspx

```

```

Const VK_STARTKEY = &H5B
Const VK_M = 77
Const KEYEVENTF_KEYUP = &H2
'WinKey down
keybd_event VK_STARTKEY, 0, 0, 0
'M key down
keybd_event VK_M, 0, 0, 0
'M key up
keybd_event VK_M, 0, KEYEVENTF_KEYUP, 0
'WinKey up
keybd_event VK_STARTKEY, 0, KEYEVENTF_KEYUP, 0

```

```

'do not minimiz form itself
'Me.WindowState = vbMaximized
'Me.WindowState = vbNormal[/b]
End Function

```

```

Sub VBE_Window()
Dim H&, W&
Application.VBE.MainWindow.Visible = True

With Application.VBE.MainWindow
.WindowState = vbext_ws_Maximize
H = .height
W = .width

.WindowState = vbext_ws_Normal

```



```

        .width = W / 2
        .left = W / 2
        .top = 0
        .height = H
        End With
    End Sub

```

```

Sub FensterMethoden()

```

```

    If 0 Then
        'Fenster der Excel session
        Application.WindowState = xlNormal
        Application.WindowState = xlMinimized
        Application.WindowState = xlNormal
        Application.Visible = False
        Application.Visible = True

        'Fenster innerhalb der Excel session
        Application.Windows(1).Activate
        Application.Windows(1).Visible = False
        Application.Windows(1).Visible = True
        Application.Windows(2).Activate
        Application.Windows(2).Visible = False
        Application.Windows(2).Visible = True
        Application.Windows(1).WindowState = xlNormal
        Application.Windows(1).WindowState = xlMinimized
    End If

```

```

    'Fenster innerhalb der Excel session (workbooks)
    Dim mywb As Workbook
    wbpPath = "C:\Users\t740698\Desktop\"
    wbName = "SmC_Download.xlsb"
    Set mywb = GetObject(wbpPath & wbName)
    'Application.mywb.Activate
    'Application.mywb.Visible = False
    'Application.mywb.Visible = True
    'Application.mywb.WindowState = xlNormal
    'Application.mywb.WindowState = xlMinimized
    mywb.Application.Visible = False
    mywb.Application.Visible = True
    mywb.Application.WindowState = xlNormal
    mywb.Application.WindowState = xlMinimized
    mywb.Application.WindowState = xlNormal
End Sub

```

```

Sub testExcelWindowFunctions()
    'Dim myWb As Workbook
    'WbPath = "C:\Users\t740698\Desktop\"
    'WbName = "SmC_Download.xlsb"
    'Set myWb = GetObject(WbPath & WbName)
    '-----
    'Call z_ExcelSessionWindowMinimized(mywb)

```

```

'Call z_ExcelSessionWindowNormal(mywb)
'Call z_ExcelSessionWindowMinimized
'Call z_ExcelSessionWindowNormal
'Call z_ExcelSessionWindowMoveAndResize(myWb, 0, 0, 900, 600)
'Call z_ExcelSessionWindowMoveAndResize(, 0, 0, 900, 600)
'Call z_ExcelWorkbookWindowMinimized(myWb)
'Call z_ExcelWorkbookWindowNormal(myWb)
'Call z_ExcelWorkbookWindowMinimizeAll
'-----

```

```

Dim mywb As Workbook
Dim wbpath As String
Dim wbname As String
wbpath = "C:\Users\t740698\Desktop\"
wbname = "SmC_Download1.xlsb"
Call z_WorkbookNewOrOpenOrActivate(wbname, wbpath, mywb)
End Sub

```

```

Sub test()
Application.Wait (Now + TimeValue("0:00:02"))
Target_X0 = 21 '(calibrate here)
Target_Y0 = 138 '(calibrate here)
'download into excel
'new position
Target_X = Target_X0 + 1019
Target_Y = Target_Y0
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:10"))
'Open
strWindowTitle = "Windows Internet Explorer provided by Syngenta"
hwnd = FindWindowHandle(strWindowTitle)
If hwnd <> 0 Then
    ' Fenster aktivieren und in den Vordergrund holen
    SetForegroundWindow hwnd
End If
'new position
Target_X = Target_X0 + 586
Target_Y = Target_Y0 + 372
SetCursorPos Target_X, Target_Y
Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
SendMouseClick (MOUSE_LEFT)
Application.Wait (Now + TimeValue("0:00:01"))

```

```

strWindowTitle = "Microsoft Excel"
hwnd = FindWindowHandle(strWindowTitle)
If hwnd <> 0 Then
    ' Fenster aktivieren und in den Vordergrund holen
    SetForegroundWindow hwnd

```

End If

```
'Target_X = Target_X0 + 567
'Target_Y = Target_Y0 + 552
'SetCursorPos Target_X, Target_Y
'Application.Wait (Now + TimeValue("0:00:01"))
'send mouse click
'SendMouseClick (MOUSE_LEFT)
'Application.Wait (Now + TimeValue("0:00:01"))
'Stop
End Sub
```

```
Private Declare Function FindWindow Lib "user32.dll" _
    Alias "FindWindowA" ( _
        ByVal lpClassName As String, _
        ByVal lpWindowName As String) As Long
Private Declare Function FindWindowEx Lib "user32.dll" _
    Alias "FindWindowExA" ( _
        ByVal hwndParent As Long, _
        ByVal hwndChildAfter As Long, _
        ByVal lpszClass As String, _
        ByVal lpszWindow As String) As Long
' Das Fensterhandle des Startbuttons ermitteln
Private Sub Command1_Click()
    Dim hwnd As Long
    Dim hwnd1 As Long

    ' Taskleiste ermitteln
    hwnd1 = FindWindow("shell_traywnd", vbNullString)

    ' Startbutton ermitteln
    hwnd = FindWindowEx(hwnd1, 0, "button", vbNullString)

    Debug.Print "Fensterhandle des Startbuttons: " & CStr(hwnd)
End Sub
```

## **File: API VBA**

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" ( _
    ByVal hwnd As Long, _
    ByVal lpOperation As String, _
    ByVal lpFile As String, _
    ByVal lpParameters As String, _
    ByVal lpDirectory As String, _
    ByVal nshowcmd As Long) As Long

Private Const SW_HIDE = 0
Private Const SW_MAXIMIZE = 3
Private Const SW_MINIMIZE = 6
Private Const SW_NORMAL = 1
```

```
Private Const SW_SHOW = 5
Private Const SW_RESTORE = 9
Private Const SW_SHOWMAXIMIZED = 3
Private Const SW_SHOWMINIMIZED = 2
Private Const SW_SHOWMINNOACTIVE = 7
Private Const SW_SHOWNA = 8
Private Const SW_SHOWNOACTIVATE = 4
Private Const SW_SHOWNORMAL = 1
```

```
Private Const ERROR_BAD_FORMAT = 11&
Private Const SE_ERR_ACCESSDENIED = 5
Private Const SE_ERR_ASSOCINCOMPLETE = 27
Private Const SE_ERR_DDEBUSY = 30
Private Const SE_ERR_DDEFAIL = 29
Private Const SE_ERR_DDETIMEOUT = 28
Private Const SE_ERR_DLLNOTFOUND = 32
Private Const SE_ERR_FNF = 2
Private Const SE_ERR_NOASSOC = 31
Private Const SE_ERR_OOM = 8
Private Const SE_ERR_PNF = 3
Private Const SE_ERR_SHARE = 26
```

```
Const HH_DISPLAY_TOPIC = &H0
Const HH_HELP_CONTEXT = &HF
```

```
Declare Function MoveWindow Lib "user32.dll" ( _
    ByVal hwnd As Long, _
    ByVal x As Long, _
    ByVal y As Long, _
    ByVal nWidth As Long, _
    ByVal nHeight As Long, _
    ByVal bRepaint As Long) As Long
```

```
Private Declare Sub Sleep Lib "kernel32" ( _
    ByVal dwMilliseconds As Long)
```

```
Private Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" ( _
    ByVal lpClassName As String, _
    ByVal lpWindowName As String) As Long
```

```
Sub MyMoveShellExecuteNotepad()
    Dim retval, rval, x, y, z As Long
    Dim AppID As Long
    retval = ShellExecute(Application.hwnd, "open", _
        "C:\Users\Benzro\Desktop\A.txt", "", _
        "", SW_NORMAL)
```

```
DoEvents
Sleep 100 'Application.Wait (Now + TimeValue("0:00:1"))
nhWnd = FindWindow(vbNullString, "A.txt - Editor")
```

```

If nhWnd <> 0 Then
    rval = MoveWindow(nhWnd, 150, 150, 500, 500, 1)
End If
Sleep 100
End Sub

```

```

Private Sub MyMoveShellNotepad()
Dim nhWnd As Long
nhWnd = Shell("C:\Windows\System32\notepad.exe", vbNormalFocus)
nhWnd = FindWindow(vbNullString, "Unbenannt - Editor")
If nhWnd <> 0 Then
    MoveWindow nhWnd, 150, 150, 500, 500, 1
End If
End Sub

```

Option Explicit

```

Private Declare Function SetCursorPos Lib "user32.dll" ( _
    ByVal x As Long, _
    ByVal y As Long) As Long

```

```

Private Declare Sub Sleep Lib "kernel32.dll" ( _
    ByVal dwMilliseconds As Long)

```

```

Private Declare Function GetCursorPos Lib "user32.dll" ( _
    ByRef lpPoint As POINTAPI) As Long

```

```

Private Type POINTAPI
    x As Long
    y As Long
End Type

```

```

Private Declare Sub mouse_event Lib "user32" _
    (ByVal dwFlags As Long, ByVal dx As Long, _
    ByVal dy As Long, ByVal cButtons As Long, _
    ByVal dwExtraInfo As Long)

```

```

Public Const MOUSE_LEFT = 0
Public Const MOUSE_MIDDLE = 1
Public Const MOUSE_RIGHT = 2

```

'Die nachfolgende Prozedur simuliert den gewünschten Mausklick.

```

Public Sub SendMausklick(ByVal mButton As Long)
    Const MOUSEEVENTF_LEFTDOWN = &H2
    Const MOUSEEVENTF_LEFTUP = &H4
    Const MOUSEEVENTF_MIDDLEDOWN = &H20
    Const MOUSEEVENTF_MIDDLEUP = &H40
    Const MOUSEEVENTF_RIGHTDOWN = &H8
    Const MOUSEEVENTF_RIGHTUP = &H10

```

```

If (mButton = MOUSE_LEFT) Then
    Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
    Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
ElseIf (mButton = MOUSE_MIDDLE) Then
    Call mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
    Call mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
Else
    Call mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
    Call mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
End If
End Sub

```

```

Sub SendMausDoppelklick(ByVal mButton As Long)
SendMausklick (mButton)
SendMausklick (mButton)
End Sub

```

```

Sub MainMausPlatzieren()
Dim myAktuellPos As POINTAPI
Dim myZielPos As POINTAPI
Dim Ziel_X, Ziel_Y As Long
Dim Aktuell_X, Aktuell_Y As Long
'aktuelle pos abfragen
GetCursorPos myAktuellPos
Aktuell_X = myAktuellPos.x
Aktuell_Y = myAktuellPos.y
MsgBox Aktuell_X & " " & Aktuell_Y
'neue pos setzen
Ziel_X = 10
Ziel_Y = 100
SetCursorPos Ziel_X, Ziel_Y
'aktuelle pos abfragen
GetCursorPos myAktuellPos
Aktuell_X = myAktuellPos.x
Aktuell_Y = myAktuellPos.y
MsgBox Aktuell_X & " " & Aktuell_Y
End Sub

```

```

Sub MainSendMousclicks()
SendMausklick (MOUSE_LEFT)
SendMausDoppelklick (MOUSE_LEFT)
End Sub

```

```

Sub ApplikationStartenUndSendKeys()
' Notepad starten und Dialog "Seite einrichten" aufrufen
Dim AppID As Long

```

```

AppID = Shell("C:\Windows\System32\notepad.exe", vbNormalFocus)
DoEvents

```

```

' NotePad aktivieren
AppActivate AppID

```

```
' Alt+d (Menü DATEI)
SendKeys "%d", True
```

```
' r (Seiten einrichten)
SendKeys "r", True
```

```
' Alt+k (Kopfzeile)
SendKeys "%k", True
```

```
' Text schreiben
SendKeys "Test-Kopfzeile", True
```

```
' Dialog beenden (OK-Schaltfläche per Alt+O auslösen)
SendKeys "{ENTER}"
```

```
' oder anstelle OK, Dialog per Alt+F4 schließen
'SendKeys "%{F4}"
```

```
End Sub
```

```
Sub EmailOrder()
'Versendet das Aktuelle Workbook als Anhang mit Windows Mail
'Erzeugt eine Warnung. Diese lässt sich ein- und ausschalten über:
Extras>Optionen>Sicherheit>Virenschutz>Warnen...
```

```
With ActiveWorkbook
.SendMail Recipients:="roland-benz@hispeed.ch", _
subject:="Testing"
End With
```

```
End Sub
```

```
'Dieses Beispiel sendet eine e-Mail über Outlook:
'Zum Anzeigen dieser Einstellungen führen Sie folgende Aktionen aus:
```

```
'Klicken Sie im Menü Extras auf Vertrauensstellungscenter.
'Klicken Sie auf Programmgesteuerter Zugriff.
```

```
Sub Mail_senden()
Dim olApp As Object
Set olApp = CreateObject("Outlook.Application")
With olApp.CreateItem(0)
'Empfänger
.Recipients.Add "roland-benz@hispeed.ch"
'Betreff
.subject = "Test-Mail"
'Nachricht
.body = "Das ist eine e-Mail" & Chr(13) & _
      "Viele Grüße..." & Chr(13) & Chr(13)
'Lesebestätigung aus
```

```

        .ReadReceiptRequested = False
        'Dateianhang
        .Attachments.Add "C:\Users\Benzro\Desktop\A.txt"
        .Send
    End With
    Set olApp = Nothing
End Sub

```

'You can copy the script and attach to any of your report  
 'but make sure that inside the VBA Editor select Tools->References and  
 'select Microsoft CDO1.21 Library before copying the script.

```

Sub SendMail()

```

```

    Dim objSession As MAPI.Session ' Local
    Dim objMessage As Message ' local
    Dim objRecip As Recipient
    On Error GoTo error_olemsg
    Dim doc As busobj.IDocument
    Dim rep As busobj.Report
    Dim DPName As String
    Dim test As Boolean

```

```

Set objSession = CreateObject("MAPI.Session")
objSession.Logon profileName:="bo_admin", NewSession:=False, showDialog:=False

```

```

If objSession Is Nothing Then
    Err.Raise 10, "MA MACRO", "must first log on; use Session->Logon"
    Exit Sub
End If

```

```

Set objMessage = objSession.Outbox.Messages.Add
If objMessage Is Nothing Then
    Err.Raise 11, "MA MACRO", "could not create a new message in the Outbox"
    Exit Sub
End If

```

```

With objMessage ' message object
    ' Substitue this with your subject
    .subject = "Resort -Monthly Report"
    ' Substitue with your the message in body part of the mail
    .Text = "The Monthly reports for " & Format(Now, "mmm") & " is attached herewith."
    For i = 1 To ThisDocument.DataProviders.Count
        If ActiveDocument Is Nothing Then
            MsgBox "NO Active Document to refresh"
        Else
            Set doc = ActiveDocument
            If Not doc.IsAddin Then

```

```

                'use this for converting to csv
            End If
        End For
    End With

```



```
DPName = "C:\" + DataProviders.Item(i).Name
test = DataProviders.Item(i).ConvertTo(boExpAsciiCSV, 1, DPName)
```

```
'use this for converting to pdf format
```

```
Else
End If
End If
Set objAttach = .Attachments.Add ' add the attachment
```

```
If objAttach Is Nothing Then
Err.Raise 12, "MA MACRO", "Unable to create new Attachmentobject"
Exit Sub
End If
```

```
With objAttach
```

```
.Name = DataProviders.Item(i).Name & ".csv"
.Source = "C:\" & DataProviders.Item(i).Name & ".csv"
```

```
End With
```

```
.Update ' update message to save attachment in MAPI system
```

```
Next i
```

```
Set objRecip = .Recipients.Add
With objRecip
objRecip.Name = ("MAILID") 'substitutue with the mailid of the recipient or groupname
objRecip.Type = CdoTo
objRecip.Resolve
End With
```

```
' use this for sending to a recipient as cc
```

```
'Set objRecip = .Recipients.Add
'With objRecip
' objRecip.Name = ("ddas")
'objRecip.Type = CdoCc
'objRecip.Resolve
'End With
```

```
.Update
' update message to save attachment in MAPI system
.Send showDialog:=False
End With
For i = 1 To ThisDocument.DataProviders.Count
Kill "C:\" & DataProviders.Item(i).Name & ".csv"
```

```
Next i
objSession.Logoff
Exit Sub
```

```
error_olemsg:
'MsgBox "Error " & Str(Err) & ": " & Error$(Err)
  Err.Raise 13, "MA MACRO", "Error " & Str(Err) & ": " & Error$(Err)
  Resume Next
```

```
End Sub
```

```
'Before you start, add a reference to the MAPI controls library.
'This is probably somewhere such as C:\Windows\System\MSMAPI32.OCX.
' Send an email message.
```

```
Public Sub SendEmail(ByVal to_name As String, ByVal _
  to_address As String, ByVal cc_name As String, ByVal _
  cc_address As String, ByVal subject As String, ByVal _
  body As String)
```

```
Dim mapi_session As MSMAPI.MAPISession
Dim mapi_messages As MSMAPI.MAPIMessages
```

```
'Debug.Print "To: " & to_name & "<" & to_address & ">"
'Debug.Print "Cc: " & cc_name & "<" & cc_address & ">"
'Debug.Print "Subject: " & subject
'Debug.Print "Body: " & body
'Debug.Print
' "====="
```

```
On Error GoTo MailError
```

```
Set mapi_session = New MSMAPI.MAPISession
With mapi_session
  .LogonUI = False
  ' Fill in username and password
  ' if necessary on this mail server.
  '.username = "username"
  '.password = "password"
  .SignOn
End With
```

```
Set mapi_messages = New MSMAPI.MAPIMessages
With mapi_messages
  .SessionID = mapi_session.SessionID
  .Compose

  .RecipIndex = 0
  .RecipDisplayName = to_name
  .RecipAddress = to_address
  .RecipType = mapToList

  .RecipIndex = 1
  .RecipDisplayName = cc_name
```

```

        .RecipAddress = cc_address
        .RecipType = mapCcList

        .AddressResolveUI = False
        .MsgSubject = subject
        .MsgNoteText = body
        .Send False
    End With

    mapi_session.SignOff
    Exit Sub

MailError:
    MsgBox Err.Description
    Exit Sub
End Sub

Sub S1()
    SendEmail "Roland Benz", "roland-benz@hispeed.ch", "", "", "Testing", "Dies ist ein Test"
End Sub

Public Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _
    (ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, _
    ByVal lpParameters As String, ByVal lpDirectory As String, _
    ByVal nshowcmd As Long) As Long

Public Const SW_HIDE = 0          ' Versteckt öffnen
Public Const SW_MAXIMIZE = 3      ' Maximiert öffnen
Public Const SW_MINIMIZE = 6      ' Minimiert öffnen
Public Const SW_NORMAL = 1
Public Const SW_RESTORE = 9
Public Const SW_SHOWMAXIMIZED = 3
Public Const SW_SHOWMINIMIZED = 2
Public Const SW_SHOWMINNOACTIVE = 7
Public Const SW_SHOWNOACTIVATE = 4

Sub DateiOeffnen()
    ' Text-Datei öffnen:
    Call ShellExecute(Application.hwnd, "open", _
        "C:\Users\Benzro\Desktop\A.txt", "", _
        "", SW_NORMAL)
    ' Word-Datei öffnen:
    'Call ShellExecute(Me.hWnd, "open", _
        "C:\MeinPfad\Mein.Doc", "", _
        "", SW_NORMAL)

    ' Excel-Datei im Hintergrund drucken:
    'Call ShellExecute(Me.hWnd, "print", _
        "C:\MeinPfad\Mein.XLS", _
        "", "", SW_HIDE)

```

' Explorer-Fenster mit einem vorgegebenen Pfad öffnen:

```
'Call ShellExecute(Me.hWnd, "explore", _  
    "", "C:\MeinPfad\", _  
    "", SW_NORMAL)
```

' Anwendung in einem bestimmten Verzeichnis ausführen, Fenster maximieren:

```
'Call ShellExecute(Me.hWnd, "Print", _  
    "C:\MeinPfad\Mein.XLS", "C:\MeinAndererPfad", _  
    "", SW_MAXIMIZE)
```

End Sub

Option Explicit

```
Private Declare Sub Sleep Lib "kernel32.dll" ( _  
    ByVal dwMilliseconds As Long)
```

```
Private Declare Function GetCursorPos Lib "user32.dll" ( _  
    ByRef lpPoint As POINTAPI) As Long
```

```
Private Declare Function SetCursorPos Lib "user32.dll" ( _  
    ByVal x As Long, _  
    ByVal y As Long) As Long
```

```
Private Declare Function GetAsyncKeyState Lib "user32.dll" (ByVal vKey As Long) As Long
```

Private Type POINTAPI

    x As Long

    y As Long

End Type

Const VK\_LBUTTON = &H1 ' Linker Mausbutton

Const VK\_RBUTTON = &H2 ' Rechter Mausbutton

Const KEYEVENTF\_KEYUP = &H2 ' Die angegebene Taste wird losgelassen

Public Sub maus\_spazieren\_Fahren()

    Dim myPos As POINTAPI

    Dim Ziel\_X As Long

    Dim Ziel\_Y As Long

    Dim L As Long

    Dim Click

    Ziel\_X = 10

    Ziel\_Y = 100

    GetCursorPos myPos

    If myPos.x >= Ziel\_X Then

        For L = myPos.x To Ziel\_X Step -1

            Sleep 5

            SetCursorPos L, myPos.y

        Next

    Else:

        For L = Ziel\_X To myPos.x

            Sleep 5

```

        SetCursorPos L, myPos.y
    Next
End If
If myPos.y >= Ziel_Y Then
    For L = myPos.y To Ziel_Y Step -1
        Sleep 5
        SetCursorPos Ziel_X, L
    Next
Else:
    For L = myPos.y To Ziel_Y
        Sleep 5
        SetCursorPos Ziel_X, L
    Next
End If
Click = GetAsyncKeyState(&H1)
MsgBox Click
End Sub

```

## **File: CCheckIfEmptyAndRemoveBlanks**

```

Sub testSpaces()
    Dim MyRange As Range
    Dim T1 As Boolean
    Dim T2 As Boolean

    'Initialize MyRange as the yello cells in Sheet2
    Set MyRange = Sheets("Sheet2").Range(Cells(1, 2), Cells(10, 6))
    'Set MyRange = Sheets("Sheet2").Range(Cells(1, 2), Cells(10, 2))
    'Set MyRange = Sheets("Sheet2").Cells(1, 2)

    'Fill MyRange with spaces and test whether MyRange is empty
    MyRange.Value = " "
    T1 = z_IsRangeEmpty("Sheet2", MyRange) 'T1=false

    'Call the function z_TrimCells to remove all spaces and test whether MyRange is empty
    Call z_TrimCells("Sheet2", MyRange)
    T2 = z_IsRangeEmpty("Sheet2", MyRange) 'T2=true
    Stop
End Sub

Function z_CellToIndex(ByRef Cell_in As Range) As Variant
    'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim CellIndexStr As String 'In R1C1 Format
    Dim CellIndexArr() As String 'Splited R1C1 Format
    Dim ColIndex As Integer
    Dim RowIndex As Integer
    Dim CellIndices(0 To 1) As Long
    'find column index
    CellIndexStr = Cell_in.Address(ReferenceStyle:=xlR1C1)
    CellIndexArr = Split(CellIndexStr, "C")
    ColIndex = CInt(CellIndexArr(1))

```

```

CellIndexArr = Split(CellIndexArr(0), "R")
RowIndex = CInt(CellIndexArr(1))
CellIndices(0) = RowIndex
CellIndices(1) = ColIndex
'Output
z_CellToIndex = CellIndices
End Function

```

```

Function z_sCellToIndex(ByRef CellIndexStr As String) As Variant
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer
Dim RowIndex As Integer
Dim CellIndices(0 To 1) As Long
'find column index
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
CellIndexArr = Split(CellIndexArr(0), "R")
RowIndex = CInt(CellIndexArr(1))
CellIndices(0) = RowIndex
CellIndices(1) = ColIndex
'Output
z_sCellToIndex = CellIndices
End Function

```

```

Function z_RangeToIndices(ByRef Rng As Range) As Variant
Dim RangeIndices(0 To 3) As Long

Dim CellsArray() As String
Dim sAddr As String
sAddr = Rng.Address(ReferenceStyle:=xlR1C1)
CellsArray = Split(sAddr, ":")

Dim CellIndicesUL() As Long
CellIndicesUL = z_sCellToIndex(CellsArray(0))

Dim CellIndicesLR() As Long
On Error GoTo RangelsCell
CellIndicesLR = z_sCellToIndex(CellsArray(1))
On Error GoTo 0

RangeIndices(0) = CellIndicesUL(0)
RangeIndices(1) = CellIndicesUL(1)
RangeIndices(2) = CellIndicesLR(0)
RangeIndices(3) = CellIndicesLR(1)

z_RangeToIndices = RangeIndices
Exit Function
RangelsCell:
CellIndicesLR = z_sCellToIndex(CellsArray(0))
Resume Next
End Function

```

```

Function z_IndicesToRange(RowUL As Long, ColUL As Long, RowDR As Long, ColDR As Long) As Range
    Dim Rng As Range
    Rng = Range(Cells(RowUL, ColUL), Cells(RowDR, ColDR))
End Function

```

```

Function z_TrimCells(Sh As String, Optional ByRef Rng As Range)
    Sheets(Sh).Activate
    If Rng Is Nothing Then
        Cells.Select
    Else
        Dim MyRngIndices() As Long
        MyRngIndices = z_RangeToIndices(Rng)
        For Row = MyRngIndices(0) To MyRngIndices(2)
            For Col = MyRngIndices(1) To MyRngIndices(3)
                With Excel.WorksheetFunction
                    Cells(Row, Col) = .Trim(.Clean(Cells(Row, Col)))
                End With
            Next Col
        Next Row
    End If
End Function

```

```

Function z_IsRangeEmpty(Sh As String, ByRef Rng As Range) As Boolean
    Sheets(Sh).Activate
    Dim MyIsEmpty As Boolean
    Dim MyRngIndices() As Long
    Dim CellValue_ij As Variant
    MyRngIndices = z_RangeToIndices(Rng)
    For Row = MyRngIndices(0) To MyRngIndices(2)
        For Col = MyRngIndices(1) To MyRngIndices(3)
            CellValue_ij = Cells(Row, Col).Value
            MyIsEmpty = VBA.IsEmpty(CellValue_ij)
        Next Col
    Next Row
    z_IsRangeEmpty = MyIsEmpty
End Function

```

## **File:**

# **Confidential SmC\_PMEC\_VBA\_EvaluationForChuck\_2** **012 3 5**

```

Public flag_no_endlessloop As Boolean

```

```

Sub SetFlag()
    flag_no_endlessloop = False
End Sub

```

```

Private Sub Worksheet_Change(ByVal Target As Range)
    If flag_no_endlessloop = True Then

```

```

Exit Sub
End If
Dim pt As PivotTable
Dim pf As PivotField
Set pt = ActiveSheet.PivotTables(1)
pt.ManualUpdate = True
flag_no_endlessloop = True
Application.EnableEvents = False
For Each pf In pt.DataFields
    If Not pf Is Nothing Then
        With pf
            .Function = xlSum
            .NumberFormat = "#,##0.00_);[Red](#,##0.00)"
        End With
    End If
Next pf
pt.ManualUpdate = False
ActiveSheet.Activate
For Col = 1 To 20
    If Cells(1, Col).ColumnWidth > 30 Then
        Cells(1, Col).ColumnWidth = 30
    End If
Next Col
flag_no_endlessloop = False
Application.EnableEvents = True
End Sub

Private Sub Workbook_Open()
    flag_no_endlessloop = False
End Sub

```

## **File: CopyPaste**

```

Public MyChoice As Integer

Sub Makro()
    For Each Mywb In Workbooks
        Mystring = Mywb.FullName
        MsgBox Mystring
    Next Mywb
End Sub

Private Sub Makro1()
    'Extras>Verweise:
    'Microsoft Forms 2.0 Object Library
    'Visual Basic for Applications
    'Microsoft Excel 12.0 Object Library
    'OLE Automation
    'Microsoft Office 12.0 Object Library

```



'DataObjekt deklarieren, neue Instanz erstellen  
Set MyData = New DataObject

'Definiere die Variablen für Workbook und Worksheet  
Mywb = 1 "I:\Buchhaltung\Kassa-, PC-, Bankbücher\Dezember 2010.xlsx"  
Myws = 1 "Dezember 2010"

'Definiere die Variable für Zeilennummer  
MyRow = Application.InputBox("Zeilennummer eingeben")  
DoEvents  
Application.Wait (Now + TimeValue("0:00:5"))

'Start der Schleife über die Zeilen i  
For i = MyRow To 100

'Feld 1: Datum, Datumsliteral in Textformat konvertieren  
MyTxt = Workbooks(Mywb).Worksheets(Myws).Cells(i, 1)  
MyTxt = CStr(MyTxt)  
MyData.SetText (MyTxt)  
MyData.PutInClipboard  
Beep  
DoEvents  
Application.Wait (Now + TimeValue("0:00:3"))

'Feld 2: Text  
Workbooks(Mywb).Worksheets(Myws).Cells(i, 2).Copy  
Beep  
DoEvents  
Application.Wait (Now + TimeValue("0:00:3"))

'Feld 3: Belegnr.  
Workbooks(Mywb).Worksheets(Myws).Cells(2, 8).Copy  
Beep  
DoEvents  
Application.Wait (Now + TimeValue("0:00:3"))

'Feld 4: Gegenkonto  
Workbooks(Mywb).Worksheets(Myws).Cells(i, 3).Copy  
Beep  
DoEvents  
Application.Wait (Now + TimeValue("0:00:3"))

'Feld 5: Kst  
Workbooks(Mywb).Worksheets(Myws).Cells(i, 4).Copy  
Beep  
DoEvents

Application.Wait (Now + TimeValue("0:00:3"))  
'Feld 6: MWST, Input kürzen  
MyTxt = Workbooks(Mywb).Worksheets(Myws).Cells(i, 5)  
MyTxt = Left(MyTxt, 3)  
MyData.SetText (MyTxt)  
MyData.PutInClipboard  
Beep  
DoEvents  
Application.Wait (Now + TimeValue("0:00:3"))

'Feld 7: Soll  
Workbooks(Mywb).Worksheets(Myws).Cells(i, 6).Copy

```

Beep
DoEvents
Application.Wait (Now + TimeValue("0:00:3"))
'Feld 8: Haben
Workbooks(Mywb).Worksheets(Myws).Cells(i, 7).Copy
Beep
DoEvents
Application.Wait (Now + TimeValue("0:00:3"))

'Fortfahren, wiederholen, halt?
message = "Zeile=" & i & ": Möchten Sie fortfahren (ja drücken), wiederholen (nein drücken) oder
abbrechen ?"
UserForm2.TextBox1.Text = message
UserForm2.Show
Antwort = MyChoice 'Globale Variable von Modul1, wird in
UserForm2>CommandButton1/2/3_Click() gesetzt!
Application.Wait (Now + TimeValue("0:00:3"))
If Antwort = 1 Then
    i = i
Elseif Antwort = 2 Then
    i = i - 1
Elseif Antwort = 3 Then
    Exit Sub
End If

Next i

End Sub

Private Sub CommandButton1_Click()
MyChoice = 1
UserForm2.Hide
End Sub

Private Sub CommandButton2_Click()
MyChoice = 2
UserForm2.Hide
End Sub

Private Sub CommandButton3_Click()
MyChoice = 3
UserForm2.Hide
End Sub

Private Sub UserForm_Click()

End Sub

```

## **File: CopyundPaste**

```
Private Sub CommandButton1_Click()
```

```

Worksheets(1).Cells(5, 2).Copy
End Sub
Sub test1()
Worksheets(1).Cells(5, 2).Copy
If Application.Wait(Now + TimeValue("0:00:10")) Then
    MsgBox "Time expired"
End If
End Sub
Sub test2()
Application.OnKey "{ESC}", MyNext
Application.OnKey "{capslock}", MyPrevious
End Sub
Public Function MyNext()
Worksheets(1).Cells(5, 2).Copy
End Function
Public Function MyPrevious()
Worksheets(1).Cells(4, 2).Copy
End Function
Sub test3()
' AppActivate kann auch den Rückgabewert der Shell-Funktion verwenden.
AnwID = Shell("C:\Program Files\Microsoft Office\Office12\WINWORD.EXE", 1) ' Microsoft Word
' starten.
AppActivate AnwID ' Microsoft Word
' aktivieren.
End Sub
Sub test4()
Dim Ergebnis, i
Ergebnis = Shell("CALC.EXE", 1) ' Rechner starten.
AppActivate Ergebnis ' Rechner aktivieren.
For i = 1 To 100 ' Zählschleife beginnen.
    SendKeys i & "{+}", True ' Tastenanschläge senden, um die
Next i ' Werte von I zu addieren.
SendKeys "=", True ' Gesamtsumme abrufen.
SendKeys "%{F4}", True ' Rechner mit ALT+F4 beenden.
End Sub
Sub test5()
AppActivate "Microsoft Word"
End Sub
Sub test6()
channelNumber = Application.DDEInitiate( _
    App:="WinWord", _
    Topic:="C:\Users\Benzro\Desktop\Dok1.DOCX")
Application.DDEExecute channelNumber, "[FILEPRINT]"
Application.DDETerminate channelNumber
End Sub
Sub test7()
channelNumber = Application.DDEInitiate( _
    App:="WinWord", _
    Topic:="C:\Users\Benzro\Desktop\Dok1.DOCX")
Set rangeToPoke = Worksheets(1).Cells(4, 2)
Application.DDEPoke channelNumber, "\StartOfDoc", rangeToPoke
Application.DDETerminate channelNumber
End Sub

```

```

Sub test8()
listArray = Application.GetCustomListContents(1)
For i = LBound(listArray, 1) To UBound(listArray, 1)
    Worksheets(3).Cells(i, 1).Value = listArray(i)
Next i
End Sub
Sub UseSpeech()
    Application.Speech.Speak "Hello"
End Sub
Sub test9()
Dim appWD As Word.Application
Set appWD = CreateObject("Word.Application")
appWD.Documents.Open ("C:\Users\Benzro\Desktop\Dok1.DOCX")
appWD.Activate
appWD.Quit
End Sub
Sub test10()
Ergebnis = Shell("Winword", 1)
AppActivate Ergebnis
SendKeys "%iy~", True
End Sub
Sub test11()
For j = 2 To 3
For i = 1 To 5
    Worksheets(1).Cells(i, j).Copy
    Worksheets(2).Paste Destination:=Worksheets(2).Cells(1, 1)
    Beep
    Application.Wait (Now + TimeValue("0:00:1"))
Next i
Next j
End Sub

```

```

Private Sub CommandButton1_Click()
MyNum = Application.InputBox("Zeilennummer eingeben")
Application.Wait (Now + TimeValue("0:00:5"))
For i = MyNum To 11
For j = 2 To 4
    DoEvents
    Worksheets(1).Cells(i, j).Copy
    Worksheets(2).Cells(2, 1) = i
    Worksheets(2).Cells(2, 2) = j
    Worksheets(2).Cells(2, 3) = Worksheets(1).Cells(i, j)
    Beep
    DoEvents
    Application.Wait (Now + TimeValue("0:00:5"))
Next j
Mldg = "Möchten Sie fortfahren(ja drücken), wiederholen(nein drücken) oder abbrechen ?"
Stil = 3
Antwort = MsgBox(Mldg, Stil)
Application.Wait (Now + TimeValue("0:00:5"))
If Antwort = vbYes Then
    i = i

```

```

ElseIf Antwort = vbNo Then
    i = i - 1
Else
    Exit Sub
End If
Next i
End Sub

```

## **File:Fit Supply to Demand 2012 with macro and CD as only reference**

'To do:

\*\*\*\*\*

'A. Make the changes in the source tables (or pivots) so that for each ratio there is only 1 line in the pivots:

'-----

'1. corn: Barley and Wheat and probably others; CostDriver-Demand inconsistent; change DEMAND source (or pivot) to Barley and Wheat

'2. facility: CostDriver-Demand; GH passive and GH active; change BOTH sources (or pivots) to GH

'3. facility: CostDriver-Demand inconsistent; for nurseries - inbreds change in BOTH SOURCES the facility to "not used"

'4. facility: Hughes-Demand inconsistent; for yield trial and nurseries-observation change in DEMAND SOURCE the facility to "not used"

'5. all attributes: fill the (Blanks) with values in the DEMAND SOURCE

'Before starting:

'-----

'1. year: set filter in Hughes pivot

'2. all attributes: remove all auto filters in the pivots

'3. set the flags to true or false in Main\_Fitting under "2. Invoke Fitting Algorithms (VBA Outputs)"

'4. set the parameters accordingly for layout changes in Main\_Fitting under "1. Define the layout parameters"

\*\*\*\*\*

Sub Main\_Fitting()

'1. Define the layout parameters

\*\*\*\*\*

'1.1 Layout Demand (FT input for Hughes input)

'-----

Dim Sh\_DemandPivot\_ForHughesInput As Worksheet

Set Sh\_DemandPivot\_ForHughesInput = Sheets("demand pivot ex FP temp - H")

Dim Col\_DemandPivot\_ForHughesInput\_A\_Country As Long

Col\_DemandPivot\_ForHughesInput\_A\_Country = z\_GetColumnIndex("country", 4, Sh\_DemandPivot\_ForHughesInput.Name)

Dim Col\_DemandPivot\_ForHughesInput\_C\_Crop As Long

Col\_DemandPivot\_ForHughesInput\_C\_Crop = z\_GetColumnIndex("crop", 4, Sh\_DemandPivot\_ForHughesInput.Name)

```

Dim Col_DemandPivot_ForHughesInput_D_Activity As Long
    Col_DemandPivot_ForHughesInput_D_Activity = z_GetColumnIndex("activity / trial type", 4,
Sh_DemandPivot_ForHughesInput.Name)
Dim Col_DemandPivot_ForHughesInput_E_Facility As Long
    Col_DemandPivot_ForHughesInput_E_Facility = z_GetColumnIndex("facility adjusted", 4,
Sh_DemandPivot_ForHughesInput.Name)
Dim Col_DemandPivot_ForHughesInput_F_SumOfTotalNoPlots As Long
    Col_DemandPivot_ForHughesInput_F_SumOfTotalNoPlots = z_GetColumnIndex("Sum of total
no. plots (or no. detailed facility ex column S)", 4, Sh_DemandPivot_ForHughesInput.Name)
Dim Col_DemandPivot_ForHughesInput_G_SumOfTotalNoPlants As Long
    Col_DemandPivot_ForHughesInput_G_SumOfTotalNoPlants = z_GetColumnIndex("Sum of total
no. plants (calculated)", 4, Sh_DemandPivot_ForHughesInput.Name)
Dim Row_DemandPivot_ForHughesInput_From As Long
    Row_DemandPivot_ForHughesInput_From = 5
Dim Row_DemandPivot_ForHughesInput_To As Long
    Row_DemandPivot_ForHughesInput_To = z_RowSize(1,
Sh_DemandPivot_ForHughesInput.Name)
'Layout DemandPivot compression
'-----
Dim Layout_DemandPivot_ForHughesInput(1 To 8) As Variant
    Layout_DemandPivot_ForHughesInput(1) = Col_DemandPivot_ForHughesInput_A_Country
    Layout_DemandPivot_ForHughesInput(2) = Col_DemandPivot_ForHughesInput_C_Crop
    Layout_DemandPivot_ForHughesInput(3) = Col_DemandPivot_ForHughesInput_D_Activity
    Layout_DemandPivot_ForHughesInput(4) = Col_DemandPivot_ForHughesInput_E_Facility
    Layout_DemandPivot_ForHughesInput(5) =
Col_DemandPivot_ForHughesInput_F_SumOfTotalNoPlots
    Layout_DemandPivot_ForHughesInput(6) =
Col_DemandPivot_ForHughesInput_G_SumOfTotalNoPlants
    Layout_DemandPivot_ForHughesInput(7) = Row_DemandPivot_ForHughesInput_From
    Layout_DemandPivot_ForHughesInput(8) = Row_DemandPivot_ForHughesInput_To

'1.2 Layout Demand (FT input for CostDriver input)
'-----
Dim Sh_DemandPivot_ForCostDriverInput As Worksheet
Set Sh_DemandPivot_ForCostDriverInput = Sheets("demand pivot ex FP temp - CD")
Dim Col_DemandPivot_ForCostDriverInput_A_Country As Long
    Col_DemandPivot_ForCostDriverInput_A_Country = z_GetColumnIndex("country", 4,
Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_C_Crop As Long
    Col_DemandPivot_ForCostDriverInput_C_Crop = z_GetColumnIndex("crop adjusted", 4,
Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_D_Activity As Long
    Col_DemandPivot_ForCostDriverInput_D_Activity = z_GetColumnIndex("activity / trial type", 4,
Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_E_Facility As Long
    Col_DemandPivot_ForCostDriverInput_E_Facility = z_GetColumnIndex("facility adjusted", 4,
Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_F_SumOfTotalNoPlots As Long
    Col_DemandPivot_ForCostDriverInput_F_SumOfTotalNoPlots = z_GetColumnIndex("Sum of total
no. plots (or no. detailed facility ex column S)", 4, Sh_DemandPivot_ForCostDriverInput.Name)
Dim Col_DemandPivot_ForCostDriverInput_G_SumOfTotalNoPlants As Long
    Col_DemandPivot_ForCostDriverInput_G_SumOfTotalNoPlants = z_GetColumnIndex("Sum of
total no. plants (calculated)", 4, Sh_DemandPivot_ForCostDriverInput.Name)

```

```

Dim Row_DemandPivot_ForCostDriverInput_From As Long
    Row_DemandPivot_ForCostDriverInput_From = 5
Dim Row_DemandPivot_ForCostDriverInput_To As Long
    Row_DemandPivot_ForCostDriverInput_To = z_RowSize(1,
Sh_DemandPivot_ForCostDriverInput.Name)
'Layout DemandPivot compression
'-----

Dim Layout_DemandPivot_ForCostDriverInput(1 To 8) As Variant
    Layout_DemandPivot_ForCostDriverInput(1) =
Col_DemandPivot_ForCostDriverInput_A_Country
    Layout_DemandPivot_ForCostDriverInput(2) = Col_DemandPivot_ForCostDriverInput_C_Crop
    Layout_DemandPivot_ForCostDriverInput(3) = Col_DemandPivot_ForCostDriverInput_D_Activity
    Layout_DemandPivot_ForCostDriverInput(4) = Col_DemandPivot_ForCostDriverInput_E_Facility
    Layout_DemandPivot_ForCostDriverInput(5) =
Col_DemandPivot_ForCostDriverInput_F_SumOfTotalNoPlots
    Layout_DemandPivot_ForCostDriverInput(6) =
Col_DemandPivot_ForCostDriverInput_G_SumOfTotalNoPlants
    Layout_DemandPivot_ForCostDriverInput(7) = Row_DemandPivot_ForCostDriverInput_From
    Layout_DemandPivot_ForCostDriverInput(8) = Row_DemandPivot_ForCostDriverInput_To

```

'1.3 Layout Supply (Hughes input)

```

'-----
Dim Sh_SupplyHughesPivot As Worksheet
Set Sh_SupplyHughesPivot = Sheets("Hughes' pivot")
Dim Col_SupplyHughesPivot_A_Country As Long
    Col_SupplyHughesPivot_A_Country = z_GetColumnIndex("Country of trial achievement", 3,
Sh_SupplyHughesPivot.Name)
Dim Col_SupplyHughesPivot_C_Crop As Long
    Col_SupplyHughesPivot_C_Crop = z_GetColumnIndex("Crop type2 (detailed)", 3,
Sh_SupplyHughesPivot.Name)
Dim Col_SupplyHughesPivot_D_Activity As Long
    Col_SupplyHughesPivot_D_Activity = z_GetColumnIndex("Type of Field activity", 3,
Sh_SupplyHughesPivot.Name)
Dim Col_SupplyHughesPivot_E_SumOfTotalNoPlots As Long
    Col_SupplyHughesPivot_E_SumOfTotalNoPlots = z_GetColumnIndex("Sum of Nb Real plots", 3,
Sh_SupplyHughesPivot.Name)
Dim Row_SupplyHughesPivot_From As Long
    Row_SupplyHughesPivot_From = 5
Dim Row_SupplyHughesPivot_To As Long
    Row_SupplyHughesPivot_To = z_RowSize(1, Sh_SupplyHughesPivot.Name)
'Layout SupplyHughesPivot compression
'-----

Dim Layout_SupplyHughesPivot(1 To 6) As Variant
    Layout_SupplyHughesPivot(1) = Col_SupplyHughesPivot_A_Country
    Layout_SupplyHughesPivot(2) = Col_SupplyHughesPivot_C_Crop
    Layout_SupplyHughesPivot(3) = Col_SupplyHughesPivot_D_Activity
    Layout_SupplyHughesPivot(4) = Col_SupplyHughesPivot_E_SumOfTotalNoPlots
    Layout_SupplyHughesPivot(5) = Row_SupplyHughesPivot_From
    Layout_SupplyHughesPivot(6) = Row_SupplyHughesPivot_To

```

'1.4 Layout Supply (CostDriver input)

```

'-----
Dim Sh_SupplyCostDriverPivot As Worksheet

```

```

Set Sh_SupplyCostDriverPivot = Sheets("Cost Driver pivot")
Dim Col_SupplyCostDriverPivot_A_Country As Long
    Col_SupplyCostDriverPivot_A_Country = z_GetColumnIndex("country", 3,
Sh_SupplyCostDriverPivot.Name)
Dim Col_SupplyCostDriverPivot_C_Crop As Long
    Col_SupplyCostDriverPivot_C_Crop = z_GetColumnIndex("crop adjusted", 3,
Sh_SupplyCostDriverPivot.Name)
Dim Col_SupplyCostDriverPivot_D_Activity As Long
    Col_SupplyCostDriverPivot_D_Activity = z_GetColumnIndex("cost driver", 3,
Sh_SupplyCostDriverPivot.Name)
Dim Col_SupplyCostDriverPivot_E_Facility As Long
    Col_SupplyCostDriverPivot_E_Facility = z_GetColumnIndex("facility adjusted", 3,
Sh_SupplyCostDriverPivot.Name)
Dim Col_SupplyCostDriverPivot_F_Units As Long
    Col_SupplyCostDriverPivot_F_Units = z_GetColumnIndex("unit for all regions", 3,
Sh_SupplyCostDriverPivot.Name)
Dim Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits As Long
    Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits = z_GetColumnIndex("Sum of no. of units
supplied", 3, Sh_SupplyCostDriverPivot.Name)
Dim Row_SupplyCostDriverPivot_From As Long
    Row_SupplyCostDriverPivot_From = 5
Dim Row_SupplyCostDriverPivot_To As Long
    Row_SupplyCostDriverPivot_To = z_RowSize(1, Sh_SupplyCostDriverPivot.Name)
'Layout SupplyCostDriverPivot compression
'-----
Dim Layout_SupplyCostDriverPivot(1 To 8) As Variant
    Layout_SupplyCostDriverPivot(1) = Col_SupplyCostDriverPivot_A_Country
    Layout_SupplyCostDriverPivot(2) = Col_SupplyCostDriverPivot_C_Crop
    Layout_SupplyCostDriverPivot(3) = Col_SupplyCostDriverPivot_D_Activity
    Layout_SupplyCostDriverPivot(4) = Col_SupplyCostDriverPivot_E_Facility
    Layout_SupplyCostDriverPivot(5) = Col_SupplyCostDriverPivot_F_Units
    Layout_SupplyCostDriverPivot(6) = Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits
    Layout_SupplyCostDriverPivot(7) = Row_SupplyCostDriverPivot_From
    Layout_SupplyCostDriverPivot(8) = Row_SupplyCostDriverPivot_To

```

'1.4 Layout Fitting (VBA Output)

'-----

```

Dim Col_Fitting_B_Country As Long
    Col_Fitting_B_Country = 2
Dim Row_Fitting_6_Crop As Long
    Row_Fitting_6_Crop = 6
Dim Row_Fitting_From As Long
    Row_Fitting_From = 8
Dim Row_Fitting_To As Long
    Row_Fitting_To = 55
Dim Col_Fitting_From As Long
    Col_Fitting_From = 3
Dim Col_Fitting_To As Long
    Col_Fitting_To = 41
'Layout Fitting compression
'-----
Dim Layout_Fitting(1 To 6) As Variant
    Layout_Fitting(1) = Col_Fitting_B_Country

```



```

Layout_Fitting(2) = Row_Fitting_6_Crop
Layout_Fitting(3) = Row_Fitting_From
Layout_Fitting(4) = Row_Fitting_To
Layout_Fitting(5) = Col_Fitting_From
Layout_Fitting(6) = Col_Fitting_To

```

```
'2. Invoke Fitting Algorithms (VBA Outputs)
```

```
*****
```

```
Dim Flag_YieldTrials As Boolean
```

```
Flag_YieldTrials = True
```

```
Dim Flag_NurseriesObservation As Boolean
```

```
Flag_NurseriesObservation = True
```

```
Dim Flag_NurseriesCrossesField As Boolean
```

```
Flag_NurseriesCrossesField = True
```

```
Dim Flag_NurseriesCrossesGreenHouses As Boolean
```

```
Flag_NurseriesCrossesGreenHouses = True
```

```
Dim Flag_NurseriesInbreds As Boolean
```

```
Flag_NurseriesInbreds = True
```

```
'Demand (FT input)
```

```
'Supply (Hughes input)
```

```
'-----
```

```
' '2.1.1 fitting yield trials
```

```
' '-----
```

```
' Dim Sh_Fitting_YieldTrials As Worksheet
```

```
' Set Sh_Fitting_YieldTrials = Sheets("fitting yield trials")
```

```
' '!Layout_Fitting(0) = Sh_Fitting_YieldTrials
```

```
' 'activity search string
```

```
' '-----
```

```
' Dim Value_DemandPivot_YieldTrials_D_Activity As String
```

```
' Value_DemandPivot_YieldTrials_D_Activity = "yield trial"
```

```
' Dim Value_SupplyHughesPivot_YieldTrials_D_Activity As String
```

```
' Value_SupplyHughesPivot_YieldTrials_D_Activity = "yield trial"
```

```
' 'facility search string
```

```
' '-----
```

```
' Dim Value_DemandPivot_YieldTrials_E_Facility As Variant
```

```
' Value_DemandPivot_YieldTrials_E_Facility = "not used"
```

```
' Dim Value_SupplyHughesPivot_YieldTrials_E_Facility As Variant
```

```
' Value_SupplyHughesPivot_YieldTrials_E_Facility = Empty
```

```
' 'invoke fitting algorithm
```

```
' '-----
```

```
' Dim Unit As String
```

```
' Unit = "plot"
```

```
' Dim Supply As String
```

```
' Supply = "Hughes Pivot"
```

```
' If Flag_YieldTrials Then
```

```
' Call z_Fitting_Clear(Sh_Fitting_YieldTrials, Layout_Fitting)
```

```
' Call z_Fitting(Sh_Fitting_YieldTrials, _
```

```
' Sh_DemandPivot_ForHughesInput, _
```

```
' Sh_SupplyHughesPivot, _
```

```
' Sh_SupplyCostDriverPivot, _
```

```
' Layout_Fitting, _
```

```
' Layout_DemandPivot_ForHughesInput, _
```

```

'         Layout_SupplyHughesPivot, _
'         Layout_SupplyCostDriverPivot, _
'         Unit, _
'         Supply, _
'         Value_DemandPivot_YieldTrials_D_Activity, _
'         Value_SupplyHughesPivot_YieldTrials_D_Activity, _
'         Value_DemandPivot_YieldTrials_E_Facility, _
'         Value_SupplyHughesPivot_YieldTrials_E_Facility)
' End If

' 2.1.2 fitting nurseries-observation
' -----
' Dim Sh_Fitting_NurseriesObservation As Worksheet
' Set Sh_Fitting_NurseriesObservation = Sheets("fitting nurseries-observation")
' !Layout_Fitting(0) = Sh_Fitting_NurseriesObservation
'   'activity search string
'   '-----
'   Dim Value_DemandPivot_NurseriesObservation_D_Activity As String
'   Value_DemandPivot_NurseriesObservation_D_Activity = "nurseries - observation"
'   Dim Value_SupplyHughesPivot_NurseriesObservation_D_Activity As String
'   Value_SupplyHughesPivot_NurseriesObservation_D_Activity = "observation"
'   'facility search string
'   '-----
'   Dim Value_DemandPivot_NurseriesObservation_E_Facility As Variant
'   Value_DemandPivot_NurseriesObservation_E_Facility = "not used"
'   Dim Value_SupplyHughesPivot_NurseriesObservation_E_Facility As Variant
'   Value_SupplyHughesPivot_NurseriesObservation_E_Facility = Empty
'   'invoke fitting algorithm
'   '-----
'   Unit = "plot"
'   Supply = "Hughes Pivot"
' If Flag_NurseriesObservation Then
'   Call z_Fitting_Clear(Sh_Fitting_NurseriesObservation, Layout_Fitting)
'   Call z_Fitting(Sh_Fitting_NurseriesObservation, _
'     Sh_DemandPivot_ForHughesInput, _
'     Sh_SupplyHughesPivot, _
'     Sh_SupplyCostDriverPivot, _
'     Layout_Fitting, _
'     Layout_DemandPivot_ForHughesInput, _
'     Layout_SupplyHughesPivot, _
'     Layout_SupplyCostDriverPivot, _
'     Unit, _
'     Supply, _
'     Value_DemandPivot_NurseriesObservation_D_Activity, _
'     Value_SupplyHughesPivot_NurseriesObservation_D_Activity, _
'     Value_DemandPivot_NurseriesObservation_E_Facility, _
'     Value_SupplyHughesPivot_NurseriesObservation_E_Facility)
' End If

' Demand (FT input)
' Supply (CostDriver input)
' -----
' 2.2.1 fitting nurseries-crosses field

```

```

'-----
Dim Sh_Fitting_NurseriesCrossesField As Worksheet
Set Sh_Fitting_NurseriesCrossesField = Sheets("fitting nurseries-crosses field")
'!Layout_Fitting(0) = Sh_Fitting_NurseriesCrossesField
'activity search string
'-----

Dim Value_DemandPivot_NurseriesCrossesField_D_Activity As String
Value_DemandPivot_NurseriesCrossesField_D_Activity = "nurseries - crosses"
Dim Value_SupplyCostDriverPivot_NurseriesCrossesField_D_Activity As String
Value_SupplyCostDriverPivot_NurseriesCrossesField_D_Activity = "nurseries - crosses"
'facility search string
'-----

Dim Value_DemandPivot_NurseriesCrossesField_E_Facility As Variant
Value_DemandPivot_NurseriesCrossesField_E_Facility = "open field"
Dim Value_SupplyCostDriverPivot_NurseriesCrossesField_E_Facility As Variant
Value_SupplyCostDriverPivot_NurseriesCrossesField_E_Facility = "open field"
'invoke fitting algorithm
'-----

Dim Unit As String
Dim Supply As String
Unit = "row"
Supply = "Cost Driver Pivot"
If Flag_NurseriesCrossesField Then
Call z_Fitting_Clear(Sh_Fitting_NurseriesCrossesField, Layout_Fitting)
Call z_Fitting(Sh_Fitting_NurseriesCrossesField, _
Sh_DemandPivot_ForCostDriverInput, _
Sh_SupplyHughesPivot, _
Sh_SupplyCostDriverPivot, _
Layout_Fitting, _
Layout_DemandPivot_ForCostDriverInput, _
Layout_SupplyHughesPivot, _
Layout_SupplyCostDriverPivot, _
Unit, _
Supply, _
Value_DemandPivot_NurseriesCrossesField_D_Activity, _
Value_SupplyCostDriverPivot_NurseriesCrossesField_D_Activity, _
Value_DemandPivot_NurseriesCrossesField_E_Facility, _
Value_SupplyCostDriverPivot_NurseriesCrossesField_E_Facility)
End If

'2.2.2 fitting nurseries-crosses GH
'-----

Dim Sh_Fitting_NurseriesCrossesGreenHouses As Worksheet
Set Sh_Fitting_NurseriesCrossesGreenHouses = Sheets("fitting nurseries-crosses GH")
'!Layout_Fitting(0) = Sh_Fitting_NurseriesCrossesGreenHouses
'activity search string
'-----

Dim Value_DemandPivot_NurseriesCrossesGreenHouses_D_Activity As String
Value_DemandPivot_NurseriesCrossesGreenHouses_D_Activity = "nurseries - crosses"
Dim Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_D_Activity As String
Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_D_Activity = "nurseries -
crosses"
'facility search string

```

```

'-----
Dim Value_DemandPivot_NurseriesCrossesGreenHouses_E_Facility As Variant
Value_DemandPivot_NurseriesCrossesGreenHouses_E_Facility = "GH" ' "GH Passive" and "GH
active"
Dim Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_E_Facility As Variant
Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_E_Facility = "GH" ' "GH Passive"
and "GH active"
'invoke fitting algorithm
'-----
Unit = "cross"
Supply = "Cost Driver Pivot"
If Flag_NurseriesCrossesGreenHouses Then
Call z_Fitting_Clear(Sh_Fitting_NurseriesCrossesGreenHouses, Layout_Fitting)
Call z_Fitting(Sh_Fitting_NurseriesCrossesGreenHouses, _
Sh_DemandPivot_ForCostDriverInput, _
Sh_SupplyHughesPivot, _
Sh_SupplyCostDriverPivot, _
Layout_Fitting, _
Layout_DemandPivot_ForCostDriverInput, _
Layout_SupplyHughesPivot, _
Layout_SupplyCostDriverPivot, _
Unit, _
Supply, _
Value_DemandPivot_NurseriesCrossesGreenHouses_D_Activity, _
Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_D_Activity, _
Value_DemandPivot_NurseriesCrossesGreenHouses_E_Facility, _
Value_SupplyCostDriverPivot_NurseriesCrossesGreenHouses_E_Facility)
End If

```

#### '2.2.3.fitting nurseries-inbreds

```

'-----
Dim Sh_Fitting_NurseriesInbreds As Worksheet
Set Sh_Fitting_NurseriesInbreds = Sheets("fitting nurseries-inbreds")
'!Layout_Fitting(0) = Sh_Fitting_NurseriesInbreds
'activity search string
'-----
Dim Value_DemandPivot_NurseriesInbreds_D_Activity As String
Value_DemandPivot_NurseriesInbreds_D_Activity = "nurseries - inbreds"
Dim Value_SupplyCostDriverPivot_NurseriesInbreds_D_Activity As String
Value_SupplyCostDriverPivot_NurseriesInbreds_D_Activity = "nurseries - inbreds"
'facility search string
'-----
Dim Value_DemandPivot_NurseriesInbreds_E_Facility As Variant
Value_DemandPivot_NurseriesInbreds_E_Facility = "not used"
Dim Value_SupplyCostDriverPivot_NurseriesInbreds_E_Facility As Variant
Value_SupplyCostDriverPivot_NurseriesInbreds_E_Facility = "not used"
'invoke fitting algorithm
'-----
Unit = "plant"
Supply = "Cost Driver Pivot"
If Flag_NurseriesInbreds Then
Call z_Fitting_Clear(Sh_Fitting_NurseriesInbreds, Layout_Fitting)
Call z_Fitting(Sh_Fitting_NurseriesInbreds, _

```

```

        Sh_DemandPivot_ForCostDriverInput, _
        Sh_SupplyHughesPivot, _
        Sh_SupplyCostDriverPivot, _
        Layout_Fitting, _
        Layout_DemandPivot_ForCostDriverInput, _
        Layout_SupplyHughesPivot, _
        Layout_SupplyCostDriverPivot, _
        Unit, _
        Supply, _
        Value_DemandPivot_NurseriesInbreds_D_Activity, _
        Value_SupplyCostDriverPivot_NurseriesInbreds_D_Activity, _
        Value_DemandPivot_NurseriesInbreds_E_Facility, _
        Value_SupplyCostDriverPivot_NurseriesInbreds_E_Facility)
End If

```

'2.2.4 fitting nurseries-observation

```

'-----
Dim Sh_Fitting_NurseriesObservation As Worksheet
Set Sh_Fitting_NurseriesObservation = Sheets("fitting nurseries-observation")
'!Layout_Fitting(0) = Sh_Fitting_NurseriesObservation
'activity search string
'-----
Dim Value_DemandPivot_NurseriesObservation_D_Activity As String
    Value_DemandPivot_NurseriesObservation_D_Activity = "nurseries - observation"
Dim Value_SupplyCostDriverPivot_NurseriesObservation_D_Activity As String
    Value_SupplyCostDriverPivot_NurseriesObservation_D_Activity = "nurseries - observation"
'facility search string
'-----
Dim Value_DemandPivot_NurseriesObservation_E_Facility As Variant
    Value_DemandPivot_NurseriesObservation_E_Facility = "not used"
Dim Value_SupplyCostDriverPivot_NurseriesObservation_E_Facility As Variant
    Value_SupplyCostDriverPivot_NurseriesObservation_E_Facility = "not used"
'invoke fitting algorithm
'-----
    Unit = "plot"
    Supply = "Cost Driver Pivot"
If Flag_NurseriesObservation Then
    Call z_Fitting_Clear(Sh_Fitting_NurseriesObservation, Layout_Fitting)
    Call z_Fitting(Sh_Fitting_NurseriesObservation, _
        Sh_DemandPivot_ForCostDriverInput, _
        Sh_SupplyHughesPivot, _
        Sh_SupplyCostDriverPivot, _
        Layout_Fitting, _
        Layout_DemandPivot_ForCostDriverInput, _
        Layout_SupplyHughesPivot, _
        Layout_SupplyCostDriverPivot, _
        Unit, _
        Supply, _
        Value_DemandPivot_NurseriesObservation_D_Activity, _
        Value_SupplyCostDriverPivot_NurseriesObservation_D_Activity, _
        Value_DemandPivot_NurseriesObservation_E_Facility, _
        Value_SupplyCostDriverPivot_NurseriesObservation_E_Facility)

```

End If

'2.2.5 fitting yield trials

'-----

Dim Sh\_Fitting\_YieldTrials As Worksheet

Set Sh\_Fitting\_YieldTrials = Sheets("fitting yield trials")

'!Layout\_Fitting(0) = Sh\_Fitting\_YieldTrials

'activity search string

'-----

Dim Value\_DemandPivot\_YieldTrials\_D\_Activity As String

Value\_DemandPivot\_YieldTrials\_D\_Activity = "yield trial"

Dim Value\_SupplyHughesPivot\_YieldTrials\_D\_Activity As String

Value\_SupplyHughesPivot\_YieldTrials\_D\_Activity = "yield trial"

'facility search string

'-----

Dim Value\_DemandPivot\_YieldTrials\_E\_Facility As Variant

Value\_DemandPivot\_YieldTrials\_E\_Facility = "not used"

Dim Value\_SupplyHughesPivot\_YieldTrials\_E\_Facility As Variant

Value\_SupplyHughesPivot\_YieldTrials\_E\_Facility = "not used"

'invoke fitting algorithm

'-----

Unit = "plot"

Supply = "Cost Driver Pivot"

If Flag\_YieldTrials Then

Call z\_Fitting\_Clear(Sh\_Fitting\_YieldTrials, Layout\_Fitting)

Call z\_Fitting(Sh\_Fitting\_YieldTrials, \_

Sh\_DemandPivot\_ForCostDriverInput, \_

Sh\_SupplyHughesPivot, \_

Sh\_SupplyCostDriverPivot, \_

Layout\_Fitting, \_

Layout\_DemandPivot\_ForCostDriverInput, \_

Layout\_SupplyHughesPivot, \_

Layout\_SupplyCostDriverPivot, \_

Unit, \_

Supply, \_

Value\_DemandPivot\_YieldTrials\_D\_Activity, \_

Value\_SupplyHughesPivot\_YieldTrials\_D\_Activity, \_

Value\_DemandPivot\_YieldTrials\_E\_Facility, \_

Value\_SupplyHughesPivot\_YieldTrials\_E\_Facility)

End If

End Sub

Public Function z\_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, \_

Optional Sh As String, Optional ByRef Wb As Workbook) As Long

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Output datatype change from Variant

Dim CellIndexStr As String 'In R1C1 Format

Dim CellIndexArr() As String 'Splited R1C1 Format

Dim ColIndex As Integer

'Activate the right Wb and Sh

On Error GoTo OptionalArgument:

Wb.Activate

```

On Error GoTo 0
'Sheets(Sh).Activate
Dim Sht As Worksheet
Set Sht = Sheets(Sh)
'find column name
Dim Cl As Range
'Set Cl = Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole)
Set Cl = Sht.Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole)
If Cl Is Nothing Then GoTo NameExpectedNotExistent
'find column index
'CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexStr = Cl.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
'Output
z_GetColumnIndex = ColIndex
Exit Function
OptionalArgument:
Resume Next
NameExpectedNotExistent:
ColIndex = 0
Stop 'only in test mode
z_GetColumnIndex = ColIndex
End Function

```

```

Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet
'Sheets(Sh).Activate
'Determine the row size
z_RowSize = If(IsEmpty(Sheets(Sh).Cells(1048576, SearchCol)), Sheets(Sh).Cells(1048576, SearchCol).End(xlUp).Row, 1048576)
End Function

```

```

Function z_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Row, Output: column with the last entry in that row
'SearchCol datatype changed from integer
'Activate the Sheet
'Sheets(Sh).Activate
'Determine the col size
z_ColSize = If(IsEmpty(Sheets(Sh).Cells(SearchRow, 16384)), Sheets(Sh).Cells(SearchRow, 16384).End(xlToLeft).Column, 16384)
End Function

```

```

Sub m_TrimCells_FP()
    Call z_TrimCells(ActiveSheet.Name, Range(Cells(1, 1), Cells(5000, 40)))

```

```

End Sub
Sub m_TrimCells_CD()
    Call z_TrimCells(ActiveSheet.Name, Range(Cells(1, 1), Cells(500, 10)))
End Sub

```

```

Private Function z_TrimCells(Sh As String, Optional ByRef Rng As Range)

```

```

    'Sheets("Sheet1").Activate
    'Call z_TrimCells("Sheet1", Range(Cells(3, 1), Cells(3, 5)))

```

```

    Sheets(Sh).Activate
    If Rng Is Nothing Then
        'select all
        Cells.Select
        Set Rng = Selection.Cells
    Else
        'take the input range
    End If

```

```

    Dim MyRngIndices() As Long
    MyRngIndices = z_RangeToIndices(Rng)

```

```

    On Error Resume Next
    For Row = MyRngIndices(0) To MyRngIndices(2)
        For Col = MyRngIndices(1) To MyRngIndices(3)
            With Excel.WorksheetFunction
                Cells(Row, Col) = .Trim(.Clean(Cells(Row, Col)))
            End With
        Next Col
    Next Row
    On Error GoTo 0

```

```

End Function

```

```

*****Get Range Indices

```

```

Private Function z_RangeToIndices(ByRef Rng As Range) As Variant

```

```

'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

    Dim RangeIndices(0 To 3) As Long
    Dim CellsArray() As String
    Dim sAddr As String
    sAddr = Rng.Address(ReferenceStyle:=xlR1C1)
    CellsArray = Split(sAddr, ":")
    Dim CellIndicesUL() As Long
    On Error GoTo RangeIsColumnOrRow
    CellIndicesUL = z_sCellToIndex(CellsArray(0))
    On Error GoTo 0
    Dim CellIndicesLR() As Long
    On Error GoTo RangeIsCell
    CellIndicesLR = z_sCellToIndex(CellsArray(1))
    On Error GoTo 0
    RangeIndices(0) = CellIndicesUL(0)
    RangeIndices(1) = CellIndicesUL(1)
    RangeIndices(2) = CellIndicesLR(0)
    RangeIndices(3) = CellIndicesLR(1)
    z_RangeToIndices = RangeIndices

```



```

Exit Function
RangelsCell:
CellIndicesLR = z_sCellToIndex(CellsArray(0))
Resume Next
RangelsColumnOrRow:
Dim RorC As String
RorC = Left(sAddr, 1)
OneOrMore = InStr(1, sAddr, ":", vbTextCompare)
'only one row or column
If OneOrMore = 0 Then
    If RorC = "C" Then
        RangeIndices(0) = 1
        RangeIndices(1) = z_sColumnToIndex(CellsArray(0))
        RangeIndices(2) = 1048534
        RangeIndices(3) = RangeIndices(0)
    ElseIf RorC = "R" Then
        RangeIndices(0) = z_sRowToIndex(CellsArray(0))
        RangeIndices(1) = 1
        RangeIndices(2) = RangeIndices(0)
        RangeIndices(3) = 16383
    Else
        Stop
    End If
'more than one row or column
Else
    If RorC = "C" Then
        RangeIndices(0) = 1
        RangeIndices(1) = z_sColumnToIndex(CellsArray(0))
        RangeIndices(2) = 1048534
        RangeIndices(3) = z_sColumnToIndex(CellsArray(1))
    ElseIf RorC = "R" Then
        RangeIndices(0) = z_sRowToIndex(CellsArray(0))
        RangeIndices(1) = 1
        RangeIndices(2) = z_sRowToIndex(CellsArray(1))
        RangeIndices(3) = 16383
    Else
        Stop
    End If
End If
z_RangeToIndices = RangeIndices
End Function

```

```

Private Function z_sCellToIndex(ByRef CellIndexStr As String) As Variant

```

```

'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

    Dim CellIndexArr() As String 'Splitted R1C1 Format

```

```

    Dim ColIndex As Integer

```

```

    Dim RowIndex As Integer

```

```

    Dim CellIndices(0 To 1) As Long

```

```

    'find column index

```

```

    CellIndexArr = Split(CellIndexStr, "C")

```

```

    ColIndex = CInt(CellIndexArr(1))

```

```

    CellIndexArr = Split(CellIndexArr(0), "R")

```

```

    RowIndex = CInt(CellIndexArr(1))

```

```

CellIndices(0) =RowIndex
CellIndices(1) = ColIndex
'Output
z_sCellToIndex = CellIndices
End Function

```

```

Private Function z_sColumnToIndex(ByRef ColIndexStrLeft As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Dim ColArray() As String
ColArray = Split(ColIndexStrLeft, "C")
z_sColumnToIndex = ColArray(1)
End Function

```

```

Private Function z_sRowToIndex(ByRef RowIndexStrUp As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Dim RowArray() As String
RowArray = Split(RowIndexStrUp, "R")
z_sRowToIndex = RowArray(1)
End Function
*****Get Range Indices

```

```

Function z_Fitting(Sh_Fitting As Worksheet, _
    Sh_DemandPivot As Worksheet, _
    Sh_SupplyHughesPivot As Worksheet, _
    Sh_SupplyCostDriverPivot As Worksheet, _
    Layout_Fitting As Variant, _
    Layout_DemandPivot As Variant, _
    Layout_SupplyHughesPivot As Variant, _
    Layout_SupplyCostDriverPivot As Variant, _
    Unit As String, _
    Supply As String, _
    Value_DemandPivot_D_Activity As String, _
    Value_SupplyPivot_D_Activity As String, _
    Value_DemandPivot_E_Facility As Variant, _
    Value_SupplyPivot_E_Facility As Variant)

```

```

'Layout_Fitting decompression
'-----

```

```

'!Dim Sh_Fitting As Worksheet
'! Sh_Fitting = Layout_Fitting(0)
Dim Col_Fitting_B_Country As Long
Col_Fitting_B_Country = Layout_Fitting(1)
Dim Row_Fitting_6_Crop As Long
Row_Fitting_6_Crop = Layout_Fitting(2)
Dim Row_Fitting_From As Long
Row_Fitting_From = Layout_Fitting(3)
Dim Row_Fitting_To As Long
Row_Fitting_To = Layout_Fitting(4)
Dim Col_Fitting_From As Long
Col_Fitting_From = Layout_Fitting(5)
Dim Col_Fitting_To As Long

```

Col\_Fitting\_To = Layout\_Fitting(6)

'Iterate through all rows of the Fitting Matrix

'-----

Dim Row\_Fitting\_iter As Long

For Row\_Fitting\_iter = Row\_Fitting\_From To Row\_Fitting\_To Step 1

'Iterate through all columns of the Fitting Matrix

'-----

Dim Col\_Fitting\_iter As Long

For Col\_Fitting\_iter = Col\_Fitting\_From To Col\_Fitting\_To Step 1

'Search Sting (Matrix Dimensions of the Fitting Table)

Dim SearchString\_Fitting(0 To 1) As Variant

SearchString\_Fitting(0) = LCase(Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_B\_Country))

SearchString\_Fitting(1) = LCase(Sh\_Fitting.Cells(Row\_Fitting\_6\_Crop, Col\_Fitting\_iter))

'for debug purposes

If LCase(SearchString\_Fitting(0)) = "france" And LCase(SearchString\_Fitting(1)) = "sunflower"

Then

'Stop 'for debug purposes

End If

'search in Demand (FT input)

'-----

Dim Cell\_DemandPivot\_SumOfTotalNo As Range

Set Cell\_DemandPivot\_SumOfTotalNo = z\_SearchIn\_DemandPivot( \_  
Sh\_DemandPivot, \_  
Layout\_DemandPivot, \_  
SearchString\_Fitting, \_  
Unit, \_  
Value\_DemandPivot\_D\_Activity, \_  
Value\_DemandPivot\_E\_Facility)

'search in Supply (Hughes input)

'-----

If Supply = "Hughes Pivot" Then

Dim Cell\_SupplyPivot\_SumOfTotalNo As Range

Set Cell\_SupplyPivot\_SumOfTotalNo = z\_SearchIn\_Supply\_HughesPivot( \_  
Sh\_SupplyHughesPivot, \_  
Layout\_SupplyHughesPivot, \_  
SearchString\_Fitting, \_  
Unit, \_  
Value\_SupplyPivot\_D\_Activity, \_  
Value\_SupplyPivot\_E\_Facility)

'search in Supply (CostDriver input)

'-----

Elseif Supply = "Cost Driver Pivot" Then

Set Cell\_SupplyPivot\_SumOfTotalNo = z\_SearchIn\_Supply\_CostDriverPivot( \_  
Sh\_SupplyCostDriverPivot, \_  
Layout\_SupplyCostDriverPivot, \_  
SearchString\_Fitting, \_

```

Unit, _
Value_SupplyPivot_D_Activity, _
Value_SupplyPivot_E_Facility)

Else
    Stop
End If

'calculate the ratio demand/supply and write it into the fitting sheet
'white=2 no demand ,red=3 demand but no supply
'-----

Call z_Calculation(Sh_Fitting, Row_Fitting_iter, Col_Fitting_iter, _
    Cell_DemandPivot_SumOfTotalNo, Cell_SupplyPivot_SumOfTotalNo)

'Set the objects to nothing
'-----

Set Cell_DemandPivot_SumOfTotalNo = Nothing
Set Cell_SupplyPivot_SumOfTotalNo = Nothing

Next Col_Fitting_iter
Next Row_Fitting_iter
End Function

```

```

Function z_Fitting_Clear(Sh_Fitting As Worksheet, _
    Layout_Fitting As Variant)

'Layout_Fitting decompression
'-----

'!Dim Sh_Fitting As Worksheet
'! Sh_Fitting = Layout_Fitting(0)
Dim Col_Fitting_B_Country As Long
    Col_Fitting_B_Country = Layout_Fitting(1)
Dim Row_Fitting_6_Crop As Long
    Row_Fitting_6_Crop = Layout_Fitting(2)
Dim Row_Fitting_From As Long
    Row_Fitting_From = Layout_Fitting(3)
Dim Row_Fitting_To As Long
    Row_Fitting_To = Layout_Fitting(4)
Dim Col_Fitting_From As Long
    Col_Fitting_From = Layout_Fitting(5)
Dim Col_Fitting_To As Long
    Col_Fitting_To = Layout_Fitting(6)

'Iterate through all rows of the Fitting Matrix
'-----

Dim Row_Fitting_iter As Long
For Row_Fitting_iter = Row_Fitting_From To Row_Fitting_To Step 1

    'Iterate through all columns of the Fitting Matrix
    '-----

    Dim Col_Fitting_iter As Long
    For Col_Fitting_iter = Col_Fitting_From To Col_Fitting_To Step 1

```

```

'Clear content and interior colour
'-----

Sh_Fitting.Cells(Row_Fitting_iter, Col_Fitting_iter).ClearContents
Sh_Fitting.Cells(Row_Fitting_iter, Col_Fitting_iter).Interior.ColorIndex = 2

Next Col_Fitting_iter
Next Row_Fitting_iter
End Function

Function z_SearchIn_DemandPivot(Sh_DemandPivot As Worksheet, _
    Layout_DemandPivot As Variant, _
    SearchString_Fitting As Variant, _
    Unit As String, _
    Value_DemandPivot_D_Activity As String, _
    Value_DemandPivot_E_Facility As Variant) As Range

'Layout_Demand decompression
'-----
'!Dim Sh_DemandPivot As Worksheet
'!Set Sh_DemandPivot = Layout_DemandPivot(0)
Dim Col_DemandPivot_A_Country As Long
    Col_DemandPivot_A_Country = Layout_DemandPivot(1)
Dim Col_DemandPivot_C_Crop As Long
    Col_DemandPivot_C_Crop = Layout_DemandPivot(2)
Dim Col_DemandPivot_D_Activity As Long
    Col_DemandPivot_D_Activity = Layout_DemandPivot(3)
Dim Col_DemandPivot_E_Facility As Long
    Col_DemandPivot_E_Facility = Layout_DemandPivot(4)
Dim Col_DemandPivot_F_SumOfTotalNoUnits As Long
    Col_DemandPivot_F_SumOfTotalNoUnits = Layout_DemandPivot(5)
Dim Col_DemandPivot_G_SumOfTotalNoPlants As Long
    Col_DemandPivot_G_SumOfTotalNoPlants = Layout_DemandPivot(6)
Dim Row_DemandPivot_From As Long
    Row_DemandPivot_From = Layout_DemandPivot(7)
Dim Row_DemandPivot_To As Long
    Row_DemandPivot_To = Layout_DemandPivot(8)

'Iterate through all rows starting at the bottom of the pivot table
'-----
Dim Row_DemandPivot
For Row_DemandPivot = Row_DemandPivot_To To Row_DemandPivot_From Step -1

    'Check the last column of the pivot table
    '-----
    Dim Cell_DemandPivot_E_Facility As Range
    Set Cell_DemandPivot_E_Facility = Sh_DemandPivot.Cells(Row_DemandPivot,
Col_DemandPivot_E_Facility)
    If LCase(Cell_DemandPivot_E_Facility.Value) = LCase(Value_DemandPivot_E_Facility) Then

        'Check the second before last column of the pivot table
        '-----

```

```

Dim Cell_DemandPivot_D_Activity As Range
Set Cell_DemandPivot_D_Activity = Sh_DemandPivot.Cells(Row_DemandPivot,
Col_DemandPivot_D_Activity)
'have not found the row therefore go one row up
If LCase(Cell_DemandPivot_D_Activity.Value) = Empty Then
    Do Until LCase(Cell_DemandPivot_D_Activity.Value) <> Empty
        Set Cell_DemandPivot_D_Activity = Cell_DemandPivot_D_Activity.Offset(-1, 0)
    Loop
End If
'have found the row therefore step in to check the next column
If LCase(Cell_DemandPivot_D_Activity.Value) = LCase(Value_DemandPivot_D_Activity) Then

    'Check the third before last column of the pivot table
    '-----
    Dim Cell_DemandPivot_C_Crop As Range
    Set Cell_DemandPivot_C_Crop = Sh_DemandPivot.Cells(Row_DemandPivot,
Col_DemandPivot_C_Crop)
    'have not found the row therefore go one row up
    If LCase(Cell_DemandPivot_C_Crop.Value) = Empty Then
        Do Until LCase(Cell_DemandPivot_C_Crop.Value) <> Empty
            Set Cell_DemandPivot_C_Crop = Cell_DemandPivot_C_Crop.Offset(-1, 0)
        Loop
    End If
    'have found the row therefore step in to check the next column
    If LCase(Cell_DemandPivot_C_Crop.Value) = LCase(SearchString_Fitting(1)) Then

        'Check the fourth before last column of the pivot table
        '-----
        Dim Cell_DemandPivot_A_Country As Range
        Set Cell_DemandPivot_A_Country = Sh_DemandPivot.Cells(Row_DemandPivot,
Col_DemandPivot_A_Country)
        'have not found the row therefore go one row up
        If LCase(Cell_DemandPivot_A_Country.Value) = Empty Then
            Do Until LCase(Cell_DemandPivot_A_Country.Value) <> Empty
                Set Cell_DemandPivot_A_Country = Cell_DemandPivot_A_Country.Offset(-1, 0)
            Loop
        End If
        'have found the row therefore step in to get the needed value
        If LCase(Cell_DemandPivot_A_Country.Value) = LCase(SearchString_Fitting(0)) Then

            'Output and exit
            '-----
            If LCase(Unit) = "plot" Or LCase(Unit) = "row" Or LCase(Unit) = "cross" Then
                Dim Cell_DemandPivot_F_SumOfTotalNoUnits As Range
                Set Cell_DemandPivot_F_SumOfTotalNoUnits =
Sh_DemandPivot.Cells(Row_DemandPivot, Col_DemandPivot_F_SumOfTotalNoUnits)
                Set z_SearchIn_DemandPivot = Cell_DemandPivot_F_SumOfTotalNoUnits
            ElseIf LCase(Unit) = "plant" Then
                Dim Cell_DemandPivot_G_SumOfTotalNoPlants As Range
                Set Cell_DemandPivot_G_SumOfTotalNoPlants =
Sh_DemandPivot.Cells(Row_DemandPivot, Col_DemandPivot_G_SumOfTotalNoPlants)
                Set z_SearchIn_DemandPivot = Cell_DemandPivot_G_SumOfTotalNoPlants
            Else

```

```

        Stop
    End If
Exit For

    End If
End If
End If
End If
Next Row_DemandPivot
End Function

```

```

Function z_SearchIn_Supply_HughesPivot(Sh_SupplyHughesPivot As Worksheet, _
    Layout_SupplyHughesPivot As Variant, _
    SearchString_Fitting As Variant, _
    Unit As String, _
    Value_SupplyHughesPivot_D_Activity As String, _
    Value_SupplyHughesPivot_E_Facility As Variant) As Range

```

```

'Layout_SupplyHughes decompression
'-----

```

```

'!Dim Sh_SupplyHughesPivot As Worksheet
'! Sh_SupplyHughesPivot = Layout_SupplyHughesPivot(0)
Dim Col_SupplyHughesPivot_A_Country As Long
    Col_SupplyHughesPivot_A_Country = Layout_SupplyHughesPivot(1)
Dim Col_SupplyHughesPivot_C_Crop As Long
    Col_SupplyHughesPivot_C_Crop = Layout_SupplyHughesPivot(2)
Dim Col_SupplyHughesPivot_D_Activity As Long
    Col_SupplyHughesPivot_D_Activity = Layout_SupplyHughesPivot(3)
Dim Col_SupplyHughesPivot_E_SumOfTotalNoPlots As Long
    Col_SupplyHughesPivot_E_SumOfTotalNoPlots = Layout_SupplyHughesPivot(4)
Dim Row_SupplyHughesPivot_From As Long
    Row_SupplyHughesPivot_From = Layout_SupplyHughesPivot(5)
Dim Row_SupplyHughesPivot_To As Long
    Row_SupplyHughesPivot_To = Layout_SupplyHughesPivot(6)

```

```

'Iterate through all rows starting at the bottom of the pivot table
'-----

```

```

Dim Row_SupplyHughesPivot As Long
For Row_SupplyHughesPivot = Row_SupplyHughesPivot_To To Row_SupplyHughesPivot_From
Step -1

```

```

'Check the last column of the pivot table
'-----

```

```

Dim Cell_SupplyHughesPivot_D_Activity As Range
Set Cell_SupplyHughesPivot_D_Activity = Sh_SupplyHughesPivot.Cells(Row_SupplyHughesPivot,
Col_SupplyHughesPivot_D_Activity)
If LCase(Cell_SupplyHughesPivot_D_Activity.Value) =
LCase(Value_SupplyHughesPivot_D_Activity) Then

```

```

'Check the second before last column of the pivot table
'-----

```

```

Dim Cell_SupplyHughesPivot_C_Crop As Range
Set Cell_SupplyHughesPivot_C_Crop = Sh_SupplyHughesPivot.Cells(Row_SupplyHughesPivot,
Col_SupplyHughesPivot_C_Crop)
'have not found the row therefore go one row up
If LCase(Cell_SupplyHughesPivot_C_Crop.Value) = Empty Then
    Do Until LCase(Cell_SupplyHughesPivot_C_Crop.Value) <> Empty
        Set Cell_SupplyHughesPivot_C_Crop = Cell_SupplyHughesPivot_C_Crop.Offset(-1, 0)
    Loop
End If
'have found the row therefore step in to check the next column
If LCase(Cell_SupplyHughesPivot_C_Crop.Value) = LCase(SearchString_Fitting(1)) Then

    'Check the third before last column of the pivot table
    '-----
    Dim Cell_SupplyHughesPivot_A_Country As Range
    Set Cell_SupplyHughesPivot_A_Country =
Sh_SupplyHughesPivot.Cells(Row_SupplyHughesPivot, Col_SupplyHughesPivot_A_Country)
'have not found the row therefore go one row up
If LCase(Cell_SupplyHughesPivot_A_Country.Value) = Empty Then
    Do Until LCase(Cell_SupplyHughesPivot_A_Country.Value) <> Empty
        Set Cell_SupplyHughesPivot_A_Country = Cell_SupplyHughesPivot_A_Country.Offset(-1,
0)
    Loop
End If
'have found the row therefore step in to get the needed value
If LCase(Cell_SupplyHughesPivot_A_Country.Value) = LCase(SearchString_Fitting(0)) Then

    'Output and exit
    '-----
    Dim Cell_SupplyHughesPivot_E_SumOfTotalNoPlots As Range
    Set Cell_SupplyHughesPivot_E_SumOfTotalNoPlots =
Sh_SupplyHughesPivot.Cells(Row_SupplyHughesPivot,
Col_SupplyHughesPivot_E_SumOfTotalNoPlots)
    Set z_SearchIn_Supply_HughesPivot = Cell_SupplyHughesPivot_E_SumOfTotalNoPlots
    Exit For

End If
End If
End If
Next Row_SupplyHughesPivot
End Function

```

```

Function z_SearchIn_Supply_CostDriverPivot(Sh_SupplyCostDriverPivot As Worksheet, _
Layout_SupplyCostDriverPivot As Variant, _
SearchString_Fitting As Variant, _
Unit As String, _
Value_SupplyCostDriverPivot_D_Activity As String, _
Value_SupplyCostDriverPivot_E_Facility As Variant) As Range

'Layout_SupplyCostDriver decompression
'-----
'!Dim Sh_SupplyCostDriverPivot As Worksheet

```



```

'! Sh_SupplyCostDriverPivot = Layout_SupplyCostDriverPivot(0)
Dim Col_SupplyCostDriverPivot_A_Country As Long
    Col_SupplyCostDriverPivot_A_Country = Layout_SupplyCostDriverPivot(1)
Dim Col_SupplyCostDriverPivot_C_Crop As Long
    Col_SupplyCostDriverPivot_C_Crop = Layout_SupplyCostDriverPivot(2)
Dim Col_SupplyCostDriverPivot_D_Activity As Long
    Col_SupplyCostDriverPivot_D_Activity = Layout_SupplyCostDriverPivot(3)
Dim Col_SupplyCostDriverPivot_E_Facility As Long
    Col_SupplyCostDriverPivot_E_Facility = Layout_SupplyCostDriverPivot(4)
Dim Col_SupplyCostDriverPivot_F_Units As Long
    Col_SupplyCostDriverPivot_F_Units = Layout_SupplyCostDriverPivot(5)
Dim Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits As Long
    Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits = Layout_SupplyCostDriverPivot(6)
Dim Row_SupplyCostDriverPivot_From As Long
    Row_SupplyCostDriverPivot_From = Layout_SupplyCostDriverPivot(7)
Dim Row_SupplyCostDriverPivot_To As Long
    Row_SupplyCostDriverPivot_To = Layout_SupplyCostDriverPivot(8)

'Iterate through all rows starting at the bottom of the pivot table
'-----
Dim Row_SupplyCostDriverPivot As Long
For Row_SupplyCostDriverPivot = Row_SupplyCostDriverPivot_To To
Row_SupplyCostDriverPivot_From Step -1

    'Check the last column of the pivot table
    '-----
    Dim Cell_SupplyCostDriverPivot_E_Facility As Range
    Set Cell_SupplyCostDriverPivot_E_Facility =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_E_Facility)
    If LCase(Cell_SupplyCostDriverPivot_E_Facility.Value) =
LCase(Value_SupplyCostDriverPivot_E_Facility) Then

        'Check the second before last column of the pivot table
        '-----
        Dim Cell_SupplyCostDriverPivot_D_Activity As Range
        Set Cell_SupplyCostDriverPivot_D_Activity =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_D_Activity)
        'have not found the row therefore go one row up
        If LCase(Cell_SupplyCostDriverPivot_D_Activity.Value) = Empty Then
            Do Until LCase(Cell_SupplyCostDriverPivot_D_Activity.Value) <> Empty
                Set Cell_SupplyCostDriverPivot_D_Activity =
Cell_SupplyCostDriverPivot_D_Activity.Offset(-1, 0)
            Loop
        End If
        'have found the row therefore step in to check the next column
        If LCase(Cell_SupplyCostDriverPivot_D_Activity.Value) =
LCase(Value_SupplyCostDriverPivot_D_Activity) Then

            'Check the third before last column of the pivot table
            '-----
            Dim Cell_SupplyCostDriverPivot_C_Crop As Range
            Set Cell_SupplyCostDriverPivot_C_Crop =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_C_Crop)

```

```

'have not found the row therefore go one row up
If LCase(Cell_SupplyCostDriverPivot_C_Crop.Value) = Empty Then
    Do Until LCase(Cell_SupplyCostDriverPivot_C_Crop.Value) <> Empty
        Set Cell_SupplyCostDriverPivot_C_Crop = Cell_SupplyCostDriverPivot_C_Crop.Offset(-1,
0)
        Loop
    End If
'have found the row therefore step in to check the next column
If LCase(Cell_SupplyCostDriverPivot_C_Crop.Value) = LCase(SearchString_Fitting(1)) Then

    'Check the fourth before last column of the pivot table
    '-----
    Dim Cell_SupplyCostDriverPivot_A_Country As Range
    Set Cell_SupplyCostDriverPivot_A_Country =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_A_Country)
    'have not found the row therefore go one row up
    If LCase(Cell_SupplyCostDriverPivot_A_Country.Value) = Empty Then
        Do Until LCase(Cell_SupplyCostDriverPivot_A_Country.Value) <> Empty
            Set Cell_SupplyCostDriverPivot_A_Country =
Cell_SupplyCostDriverPivot_A_Country.Offset(-1, 0)
        Loop
    End If
    'have found the row therefore step in to get the needed value
    If LCase(Cell_SupplyCostDriverPivot_A_Country) = LCase(SearchString_Fitting(0)) Then

        'Output and exit
        '-----
        Dim Cell_SupplyCostDriverPivot_F_Units As Range
        Set Cell_SupplyCostDriverPivot_F_Units =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot, Col_SupplyCostDriverPivot_F_Units)
        If LCase(Unit) = LCase(Cell_SupplyCostDriverPivot_F_Units.Value) Then
            Dim Cell_SupplyCostDriverPivot_G_SumOfTotalNoUnits As Range
            Set Cell_SupplyCostDriverPivot_G_SumOfTotalNoUnits =
Sh_SupplyCostDriverPivot.Cells(Row_SupplyCostDriverPivot,
Col_SupplyCostDriverPivot_G_SumOfTotalNoUnits)
            Set z_SearchIn_Supply_CostDriverPivot =
Cell_SupplyCostDriverPivot_G_SumOfTotalNoUnits
            Exit For
        Else
            Stop 'error: wrong unit in pivot table
        End If
    End If
End If
End If
End If
Next Row_SupplyCostDriverPivot
End Function

Function z_Calculation(Sh_Fitting As Worksheet, Row_Fitting_iter As Long, Col_Fitting_iter As Long, _
Cell_DemandPivot_SumOfTotalNo As Range, Cell_SupplyPivot_SumOfTotalNo As
Range)

```

'calculate the ratio demand/supply and write it into the fitting sheet

'black=1,white=2,red=3,green=4,blue=5,yellow=6,brown=53

-----

'Demand found

If Not Cell\_DemandPivot\_SumOfTotalNo Is Nothing Then

    'Supply found

    If Not Cell\_SupplyPivot\_SumOfTotalNo Is Nothing Then

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter) = \_

        Cell\_DemandPivot\_SumOfTotalNo.Value / Cell\_SupplyPivot\_SumOfTotalNo.Value \* 100

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Interior.ColorIndex = 2

    'Supply not found

    Else

        'Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Value = "S-NA"

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Interior.ColorIndex = 5

    End If

'Demand not found

Else

    'Supply found

    If Not Cell\_SupplyPivot\_SumOfTotalNo Is Nothing Then

        'Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Value = "D-NA"

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Interior.ColorIndex = 53

    'Supply not found

    Else

        'Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Value = "S&D-NA"

        Sh\_Fitting.Cells(Row\_Fitting\_iter, Col\_Fitting\_iter).Interior.ColorIndex = 2

    End If

End If

End Function

## **File: LayoutCheck**

Enum myenum

a\_ = 1: b\_ = 2: c\_ = 3

End Enum

Sub s1()

Dim layout\_exp() As Variant

Dim log As String

'store the expected layout of the source file

layout\_exp = Array("t1", a\_, "t2", b\_, \_  
                    "t3", c\_)

'get the actual column/layout information from the source file

text\_act = "t1" 'find layout\_exp(0 to end step 2) in the source file

'if not found write out into the logfile and iterate

col\_act = 1 'if found determine the column number

'check whether actual and expected layout of the source file do match

```

'check the column names
For i = LBound(layout_exp) To UBound(layout_exp) Step 2
If text_act = layout_exp(i) Then
    'write int the logfile
    log = "OK!"
Else
    'write int the logfile
    log = "no match"
End If
Next i

'check the column numbers
For i = LBound(layout_exp) + 1 To UBound(layout_exp) Step 2
If col_act = layout_exp(i) Then
    'write int the logfile
    log = "OK!"
Else
    'write int the logfile
    log = "no match"
End If
Next i

End Sub

```

## **File: MapFunctions**

```

Sub testMapFkt()
'Clear the sheet Log1
Sheets("Log1").Activate
Cells.Select
Selection.ClearContents
Stop
'Clear the yello cells in Sheet1
Sheets("Sheet1").Activate
Range(Cells(2, 3), Cells(22, 14)).ClearContents
Stop
'Define the Attributes from the source sheet Sh_from=Sheets2
Dim ColNames_from As Variant
ColNames_from = Array("Attr1", "Attr3", "Attr2", "Attr7", _
    "Attr9", "Attr5", "Attr8", "Attr4", "Attr6")
'Define the attributes from the target sheet Sh_to=Sheets1
'Even though the attribute names in Sh_from and Sh_to may be called differently they must
'refer to the same attribute and the have to be in the same order in both arrays!!!!
Dim ColNames_to As Variant
ColNames_to = Array("Attribute1", "Attribute3", "Attribute2", "Attribute7", _
    "Attribute9", "Attribute5", "Attribute8", "Attribute4", "Attribute6")
Call z_ShMapColumns("Sheet2", "Key1", ColNames_from, "Sheet1", "Key", ColNames_to, "Log1")
Stop
End Sub

```

```

Function z_ShMapColumns(Sh_from As String, ColName_Key_from As String, ByRef ColNames_from
As Variant, _
    Sh_to As String, ColName_Key_to As String, ByRef ColNames_to As Variant, _
    Sh_log As String, Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Application.ScreenUpdating = False
'Start time measuring
Dim Start As Date: Dim Duration As Date
Start = Now()

'create a matrix with column names and indexes
MapMatrix = MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from,
ColNames_from, Sh_to, ColNames_to)

'Find the column index of the KeyName in "Sh_from"
Dim ColIndex_Key_from As Long
ColIndex_Key_from = z_GetColumnIndex(ColName_Key_from, 1, Sh_from)

'Find the column index of the KeyName in "Sh_to"
Dim ColIndex_Key_to As Long
ColIndex_Key_to = z_GetColumnIndex(ColName_Key_to, 1, Sh_to)

'Determine the row size in Sh_to
Sheets(Sh_to).Activate
RowSize_to = z_RowSize(ColIndex_Key_to, Sh_to)

'Select the range in the column Key_to
Sheets(Sh_to).Activate
Range(Cells(2, ColIndex_Key_to), Cells(RowSize_to, ColIndex_Key_to)).Select

'Iterate through the rows with "rcheck" = Pidentifier
Dim ilog As Long
ilog = 2
For Each ValueInCol_Key_to In Selection.Cells
    'if ValueInCol_Key_to is found in Sh_from then perform the mapping
    If Not Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to,
LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns with the help of the MapMatrix
        For j = LBound(MapMatrix) To UBound(MapMatrix) Step 1
            'read the indices
            ColIndex_from_j = MapMatrix(j, 1)
            ColIndex_to_j = MapMatrix(j, 3)
            'map
            ValueInCol_Key_to.Offset(0, (ColIndex_to_j) - ColIndex_Key_to).Value = _
                Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to, _
                LookAt:=xlWhole).Offset(0, (ColIndex_from_j) - ColIndex_Key_from)
        Next j
    Else
        'write not found ValueInCol_Key_to in Sh_from into Sh_log
        Sheets(Sh_log).Cells(ilog, 2) = ValueInCol_Key_to
        Sheets(Sh_log).Cells(ilog, 4) = " not found, map them from another source file Sh_from"
        ilog = ilog + 1
    End If
End For

```

Next

'In case the mapping has changed the row height

Sheets(Sh\_to).Activate

Cells.Select

Selection.Rows.RowHeight = 15

Application.ScreenUpdating = True

'Write the durations into the logfile

Duration = Now() - Start

Sheets(Sh\_log).Cells(iLog + 2, 1) = "Duration" & CStr(Duration)

End Function

Function z\_ChkColExistence(Sh As String, ByRef ColNames As Variant, Sh\_log As String) As Boolean

'The column existence check is assumed to find all column names at the beginning

z\_ChkColExistence = True

'Determine the column indices of the ColNames array in Sh

Dim ColName\_i As String

ReDim Matrix\_ColNameColIndex(0 To UBound(ColNames), 0 To 1) As Variant

'iterate through the array

For i = LBound(ColNames\_from) To UBound(ColNames) Step 1

ColName\_i = CStr(ColNames(i))

Matrix\_ColNameColIndex(i, 0) = ColName\_i

Matrix\_ColNameColIndex(i, 1) = z\_GetColumnIndex(ColName\_i, 1, Sh)

Debug.Print Matrix\_ColNameColIndex(i, 0) & " " & Matrix\_ColNameColIndex(i, 1)

If Matrix\_ColNameColIndex(i, 1) = 0 Then

'write errors into the logfile

Sheets(Sh\_log).Cells(iLog, 2) = "ColName: "

Sheets(Sh\_log).Cells(iLog, 3) = Matrix\_ColNameColIndex(i, 0)

Sheets(Sh\_log).Cells(iLog, 4) = "not found in " & Sh

iLog = iLog + 1

'The column existence check has detected an unfound column name

z\_ChkColExistence = False

Next i

End Function

Function MakeMatrix\_Shfrom\_ColNameColIndex\_Shto\_ColNameColIndex(Sh\_from As String, ByRef

ColNames\_from As Variant, \_

Sh\_to As String, ByRef ColNames\_to As Variant) As Variant

'Determine the column indices in Sh and Sh\_new,

Dim ColName\_from\_i As String

Dim ColName\_to\_i As String

ReDim Matrix\_ColName1Index1\_ColName2Index2(0 To UBound(ColNames\_from), 0 To 3) As

Variant

For i = LBound(ColNames\_from) To UBound(ColNames\_from) Step 1

ColName\_from\_i = CStr(ColNames\_from(i))

Matrix\_ColName1Index1\_ColName2Index2(i, 0) = ColName\_from\_i

Matrix\_ColName1Index1\_ColName2Index2(i, 1) = z\_GetColumnIndex(ColName\_from\_i, 1,

Sh\_from)

ColName\_to\_i = CStr(ColNames\_to(i))

Matrix\_ColName1Index1\_ColName2Index2(i, 2) = ColName\_to\_i

```

        Matrix_ColName1Index1_ColName2Index2(i, 3) = z_GetColumnIndex(ColName_to_i, 1, Sh_to)
        Debug.Print Matrix_ColName1Index1_ColName2Index2(i, 0) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 1) _
        & " " & Matrix_ColName1Index1_ColName2Index2(i, 2) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 3)
    Next i
    MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex =
Matrix_ColName1Index1_ColName2Index2
End Function

```

```

Public Function z_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, _
    Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Output datatype change from Variant

```

```

Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer

```

```

'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
Sheets(Sh).Activate

```

```

'find column name
On Error GoTo NameExpectedNotExistent:
Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole).Select
On Error GoTo 0
'find column index
CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
'Output
z_GetColumnIndex = ColIndex
Exit Function

```

```

OptionalArgument:
Resume Next

```

```

NameExpectedNotExistent:
    ColIndex = 0
    z_GetColumnIndex = ColIndex

```

```

End Function

```

```

Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As
Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet

```

```

Sheets(Sh).Activate
'Determine the row size
z_RowSize = If(IsEmpty(Cells(1048576, SearchCol)), Cells(1048576, SearchCol).End(xlUp).Row,
1048576)
End Function

```

## File: myfile

```

Sub test()
With Worksheets("Tabelle1")
.Rows(1).Font.Bold = True
.Range("a1:d1").Value = _
    Array("Name", "Full Name", "Title", "Installed")
For i = 1 To AddIns.Count
    .Cells(i + 1, 1) = AddIns(i).Name
    .Cells(i + 1, 2) = AddIns(i).FullName
    .Cells(i + 1, 3) = AddIns(i).Title
    .Cells(i + 1, 4) = AddIns(i).Installed
Next
.Range("a1").CurrentRegion.Columns.AutoFit
End With
End Sub

```

```

Sub FormatiereBereich()
With Worksheets("Tabelle1").Range("F1:H10")
.Value = 30
.Font.Bold = True
.Interior.Color = RGB(255, 255, 0)
End With
End Sub

```

```

Sub MeineEingabe()
With Workbooks("Mappe1").Worksheets("Tabelle1").Cells(1, 10)
.Formula = "=SQRT(50)"
With .Font
.Name = "Arial"
.Bold = True
.Size = 8
End With
End With
End Sub

```

```

Sub speichern()
ActiveWorkbook.SaveAs Filename:="E:\Documents and Settings\Roli\Desktop\myfile"
End Sub

```

```

Sub ShowFolderList(folderspec)
Dim fs, f, f1, fc, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(folderspec)

```



```

Set fc = f.Files
For Each f1 In fc
    s = s & f1.Name
    s = s & vbCrLf
Next
MsgBox s
End Sub

Sub ShowDriveList()
    Dim fs, d, dc, s, n
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set dc = fs.Drives
    For Each d In dc
        s = s & d.DriveLetter & " - "
        If d.DriveType = Remote Then
            n = d.ShareName
        Else
            n = d.VolumeName
        End If
        s = s & n & vbCrLf
    Next
    MsgBox s
End Sub

```

## File: OleSteuerelemente

```

Private Sub GetUserName1()
    UserForm1.Show
End Sub

Sub test1()
    Worksheets(1).OLEObjects.Add "Forms.CommandButton.1", _
    Left:=10, Top:=10, Height:=20, Width:=100
End Sub

Sub test3()
    Worksheets(1).OLEObjects("CommandButton7"). _
    Object.Caption = "run me now"
    Worksheets(1).OLEObjects("CommandButton7").Left = 10
    Worksheets(1).OLEObjects("CommandButton7").Top = 10
    Worksheets(1).OLEObjects("CommandButton7").Height = 100
    Worksheets(1).OLEObjects("CommandButton7").Width = 100
End Sub

Sub test2()
    Set t = Worksheets(1).OLEObjects("CommandButton7").Object
    t.Caption = "test"
    Set t = Worksheets(1).CommandButton7.TopLeftCell
    With ActiveWindow
        .ScrollRow = t.Row
        .ScrollColumn = t.Column
    End With

```

```

End Sub
Sub test4()
Worksheets(1).OLEObjects.Add "Forms.TextBox.1", _
    Left:=10, Top:=10, Height:=20, Width:=100
End Sub
Sub test5()
Dim tb, i
i = 3
tb = "TextBox" & CStr(i)
Set TextBox = Worksheets(1).OLEObjects(tb).Object
TextBox.Text = "Hell"
End Sub

Sub Schaltfläche1_KlickenSieAuf()
Worksheets(1).OLEObjects.Add "Forms.CommandButton.1", _
    Left:=10, Top:=10, Height:=20, Width:=100

End Sub

Private Sub ComboBox1_Change()

End Sub

Private Sub CommandButton1_Click()

End Sub

Private Sub Label1_Click()

End Sub

Private Sub UserForm_Click()

End Sub

```

## **File: OwnSmCReport Part2**

```

Public Function z_OpenAndActivateWb(wbname As String, wbpath As String, ByRef Wb As
Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Objective: Open the Workbook RD.xlsb if not already open and activate it.
'look if wbName is existent in workbooks list
Dim i As Long
For i = Workbooks.Count To 1 Step -1
    If Workbooks(i).Name = wbname Then Exit For
Next
'if wbName is existent in workbooks list, then i<>0-> activate workbook
'if wbName is not existent then i=0-> open workbook, activate workbook
If i <> 0 Then

```

```

    Set Wb = VBA.Interaction.GetObject(wbpath & wbname)
    Wb.Activate
Else
    Set Wb = Workbooks.Open(wbpath & wbname)
    Wb.Activate
End If
End Function

```

```

Function z_WorkbookNewOrOpenOrActivate(wbname As String, wbpath As String, ByRef Wb As
Workbook)

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 19.10.2011

'Objective: Open the Workbook RD.xlsb if not already open and activate it.

'look if wbName is existent in workbooks list

Dim i As Long

For i = Workbooks.Count To 1 Step -1

    If Workbooks(i).Name = wbname Then Exit For

Next

'if wbName is existent in workbooks list, then i<>0-> activate workbook

'if wbName is not existent then i=0-> open workbook, activate workbook

If i <> 0 Then

    Set Wb = VBA.Interaction.GetObject(wbpath & wbname)

    Wb.Activate

Else

    On Error GoTo NewWB

    Set Wb = Workbooks.Open(wbpath & wbname)

    Wb.Activate

    On Error GoTo 0

End If

Exit Function

NewWB:

Set Wb = Workbooks.Add

Dim FileFormatValue As Integer

If wbname <> Empty Then

    Select Case LCase(Right(wbname, Len(wbname) - InStrRev(wbname, ".", 1)))

        Case "xls": FileFormatValue = 56

        Case "xlsx": FileFormatValue = 51

        Case "xlsm": FileFormatValue = 52

        Case "xlsb": FileFormatValue = 50

        Case Else: FileFormatValue = 0

    End Select

End If

Wb.SaveAs Filename:=wbpath & wbname, FileFormat:=FileFormatValue

End Function

```

Sub z_ShNewFlatValueCopy(Sh As String, Sh_new As String, Optional Where As String, Optional ByRef
Sh_Ref As Worksheet)

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Name of the Sh to copy, Name of the new Sh\_new, where to place the new Sh\_new,

' before or after some Sh\_Ref, at the begin or the end

    Call z\_ShAdd(Where, Sh\_Ref)

```

On Error Resume Next
ActiveSheet.Name = Sh_new
If Err.Number <> 0 Then
    Application.DisplayAlerts = False
    ActiveSheet.Delete
    Application.DisplayAlerts = True
    Sheets(Sh_new).Cells.ClearContents
End If
On Error GoTo 0
Sheets(Sh).Cells.Copy
Sheets(Sh_new).Select
Sheets(Sh_new).Range("A1").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
Sheets(Sh_new).Columns("A:GA").ColumnWidth = 20
End Sub

Sub z_ShDelete(Sh As String, Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
Wb.Activate
Application.DisplayAlerts = False
Sheets(Sh).Delete
Application.DisplayAlerts = True
End Sub

Function z_ExcelSessionWindowNormal(Optional ByRef Wb As Workbook)
'Fenster der Excel session
If Wb Is Nothing Then
    Application.WindowState = xlNormal
Else
    Wb.Application.WindowState = xlNormal
    Wb.Activate
End If
End Function

Sub z_ExcelSessionWindowMoveAndResize(Optional ByRef Wb As Workbook, _
    Optional top As Variant, Optional left As Variant, _
    Optional width As Variant, Optional height As Variant)
If Wb Is Nothing Then
    If top <> Empty Then
        Application.top = top
    End If
    If left <> Empty Then
        Application.left = left
    End If
    If width <> Empty Then
        Application.width = width
    End If
    If height <> Empty Then
        Application.height = height
    End If
Else
    If top <> Empty Then

```

```

        Wb.Application.top = CInt(top)
    End If
    If left <> Empty Then
        Wb.Application.left = CInt(left)
    End If
    If width <> Empty Then
        Wb.Application.width = CInt(width)
    End If
    If height <> Empty Then
        Wb.Application.height = CInt(height)
    End If
End If
End Sub

```

Function z\_ShNew(Sh As String, Optional Where As String, Optional ByRef Sh\_Ref As Worksheet, Optional ByRef Wb As Workbook)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Name of the new Sh, where to place the new Sh, before or after some Sh\_Ref, at the begin or the end

On Error Resume Next

WorksheetExists = (Sheets(Sh).Name <> "")

On Error GoTo 0

If WorksheetExists = False Then

    Call z\_ShAdd(Where, Sh\_Ref)

    ActiveSheet.Name = Sh 'Worksheets.Add(Before:=Worksheets(1)).Name = Sh

End If

'Clear contents

Sheets(Sh).Activate

ActiveSheet.Cells.Select

Selection.ClearContents

'Format

Sheets(Sh).Columns.ColumnWidth = 20

Sheets(Sh).Rows.RowHeight = 15

End Function

Function z\_ShAdd(Optional Where As String, Optional ByRef Sh\_Ref As Worksheet)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: where to place the added Sh, before or after some Sh\_Ref, at the begin or the end

If Not Sh\_Ref Is Nothing Then

    If Where = ("Before:=") Then

        Sheets.Add Sh\_Ref

    Elseif Where = ("After:=") Then

        Sheets.Add , Sh\_Ref

    Else

        Sheets.Add Before:=Sheets(1)

    End If

Else

    If Where = ("End") Then

        Sheets.Add After:=Sheets(Sheets.Count)

    Elseif Where = ("Begin") Then

        Sheets.Add Before:=Sheets(1)

```

Else
    Sheets.Add Before:=Sheets(1)
End If
End If
End Function

```

```

Public Function z_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, _
    Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Output datatype change from Variant
Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer
'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
Sheets(Sh).Activate
'find column name
On Error GoTo NameExpectedNotExistent:
Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole).Select
On Error GoTo 0
'find column index
CellIndexStr = ActiveCell.Address(RangeStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
'Output
z_GetColumnIndex = ColIndex
Exit Function
OptionalArgument:
Resume Next
NameExpectedNotExistent:
    ColIndex = 0
    z_GetColumnIndex = ColIndex
End Function

```

```

Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As
Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet
Sheets(Sh).Activate
'Determine the row size
z_RowSize = If(IsEmpty(Cells(1048576, SearchCol)), Cells(1048576, SearchCol).End(xlUp).Row,
1048576)
End Function

```

```

Function z_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As
Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011

```

```

'Input: Row, Output: column with the last entry in that row
'SearchCol datatype changed from integer
'Activate the Sheet
Sheets(Sh).Activate
'Determine the col size
z_ColSize = If(IsEmpty(Cells(SearchRow, 16384)), Cells(SearchRow, 16384).End(xlToLeft).Column,
16384)
End Function

```

```

Function z_CopyRange(Sh_from As String, Range_from As Range, Sh_to As String, Cell_to As Range)

End Function

```

```

Function z_CopyRow(Sh_from As String, Row_from As Long, Sh_to As String, Row_to As Long)
Sheets(Sh_from).Activate
Sheets(Sh_from).Rows(Row_from).Select
Selection.Copy
Sheets(Sh_to).Activate
Sheets(Sh_to).Rows(Row_to).Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
End Function

```

```

Function z_InsertRow(Sh_from As String, Row_from As Long, Sh_to As String, Row_to As Long)
Sheets(Sh_from).Activate
Sheets(Sh_from).Rows(Row_from).Select
Selection.Copy
Sheets(Sh_to).Activate
Sheets(Sh_to).Rows(Row_to).Select
Selection.Insert Shift:=xlDown
End Function

```

```

Function z_CopyColumn(Sh_from As String, Col_from As Long, Sh_to As String, Col_to As Long)
Sheets(Sh_from).Activate
Sheets(Sh_from).Columns(Col_from).Select
Selection.Copy
Sheets(Sh_to).Activate
Sheets(Sh_to).Columns(Col_to).Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
End Function

```

```

Function z_InsertColumn(Sh_from As String, Col_from As Long, Sh_to As String, Col_to As Long)
Sheets(Sh_from).Activate
Sheets(Sh_from).Columns(Col_from).Select
Selection.Copy
Sheets(Sh_to).Activate
Sheets(Sh_to).Columns(Col_to).Select
Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
End Function

```

```

Function z_InsertEmptyCols(Sh As String, NofCols As Integer, Col_to As Long)
Sheets(Sh).Columns(Col_to).Select

```

```

    For Iter = 1 To NofCols
        Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
    Next Iter
End Function

Function z_InsertEmptyRows(Sh As String, NofRows As Integer, Row_to As Long)
    Sheets(Sh).Rows(Row_to).Select
    For Iter = 1 To NofRows
        Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
    Next Iter
End Function

Function z_ClearRowContents(Sh As String, Row As Long)
    Sheets(Sh).Rows(Row).Select
    Selection.ClearContents
End Function

Function z_ClearColContents(Sh As String, Col As Long)
    Sheets(Sh).Columns(Col).Select
    Selection.ClearContents
End Function

Function z_DeleteRowsOfNotEmptyCells(Sh As String, Col As Long, FirstRow As Long)
    RowSize = z_RowSize(1, Sh)
    For Row = FirstRow To RowSize
        Sheets(Sh).Cells(Row, Col).EntireRow.Select
        If Sheets(Sh).Cells(Row, Col) <> Empty Then
            Sheets(Sh).Cells(Row, Col).EntireRow.Delete
            Row = Row - 1
        End If
    Next Row
End Function

Function z_DeleteRows(Sh As String, Row_from As Long, Row_to As Long)
    Cells(Row_from, 1).Select
    For Row = Row_from To Row_to
        Sheets(Sh).Rows(Row_from).EntireRow.Delete
    Next Row
End Function

Sub test23()
    Call z_DeleteRows("RD_MasterDataSet_1", 5, 7)
End Sub

Function z_Sort(Sh As String, Col1 As Long, Col2 As Long)
    RowSize = z_RowSize(1, Sh)
    ColSize = z_ColSize(1, Sh)
    Sheets(Sh).Sort.SortFields.Clear
    Sheets(Sh).Sort.SortFields.Add Key:=Range(Cells(2, Col1), Cells(RowSize, Col1)), _
        SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    Sheets(Sh).Sort.SortFields.Add Key:=Range(Cells(2, Col2), Cells(RowSize, Col2)), _
        SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets(Sh).Sort

```



```

        .SetRange Range(Cells(1, 1), Cells(RowSize, ColSize))
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
End Function

```

```

Function z_WorkbookSave(Wb As Workbook)
    Wb.Save
End Function

```

```

Function z_LastWrittenRow(Optional Sh As String, Optional StopAtCol As Long, Optional ByRef Wb As
Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 11.10.2011
'Input: All Columns, Output: last written column
'FirstEmptyCol = z_FirstEmptyCol()
Dim RowSize_max As Long: RowSize_max = 0
Dim RowSize_col As Long: RowSize_col = 0
Dim AllCols As Long: AllCols = 16384
Dim Col As Long
'Optional input
If StopAtCol = 0 Then
    StopAtCol = AllCols
End If
For Col = 1 To StopAtCol
    RowSize_col = z_RowSize(Col, Sh)
    If RowSize_col > RowSize_max Then
        RowSize_max = RowSize_col
    End If
Next Col
z_LastWrittenRow = RowSize_max
End Function

```

```

Function z_MoveSheetFromWb1ToWb2(Sh_from As String, Wb_from As Workbook, Wb_to As
Workbook, Where As String)
    Windows(Wb_from.Name).Activate
    Sheets(Sh_from).Activate
    Sheets(Sh_from).Move Before:=Wb_to.Sheets(1)
End Function

```

```

Function z_MoveWbSheetsIntoAnotherWb(Wb_ref As Workbook)
    Dim Wb As Workbook
    For Each Wb In Wb_ref.Application.Workbooks ' Workbooks
        'move all Windows to Wb_ref
        If Wb.Name <> Wb_ref.Name Then
            If Wb.Name <> "PERSONAL.XLSB" Then
                If Wb.Name <> "API_VBA_GenerateSmartchoiceReports.xlsb" Then
                    Windows(Wb.Name).Activate
                    Sheets(left(Wb.Name, 31)).Select
                    Sheets(left(Wb.Name, 31)).Move After:=Wb_ref.Sheets(Sheets.Count)
                End If
            End If
        End If
    Next Wb
End Function

```

```

        End If
    End If
End If
Next
'For Each Wb In Wb_ref.Application.Workbooks
'    If Wb.Name = "PERSONAL.XLSB" Then
'        Windows(Wb.Name).Visible = False
'    End If
'Next
End Function

```

```

Sub test()
Dim ColStart As Long
ColStart = z_ColSize(1, "RD_MasterDataSet_5") + 1
Call z_AddColNames(Array("Test1", "Test2"), "RD_MasterDataSet_5", 1)
End Sub

```

```

Function z_AddColNames(ByRef ColNames As Variant, Optional Sh As String, Optional Row As Integer
= 1, Optional Col_Start As Long, Optional ByRef Wb As Workbook)
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
If Col_Start = 0 Then
    Size_ColNames = UBound(ColNames)
    For j = Col_Start To Size_ColNames
        Cells(Row, j + 1) = ColNames(j)
    Next j
Else
    Size_ColNames = UBound(ColNames) + Col_Start
    For j = Col_Start To Size_ColNames
        Cells(Row, j) = ColNames(j - Col_Start)
    Next j
End If
Cells.Select
'Selection.EntireColumn.AutoFit
Selection.ColumnWidth = 20
End Function

```

```

Sub z_ChgFmt_CostCols(Sh As String, FromCol As Long, ToCol As Long)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 27.9.2011
Sheets(Sh).Select
RowSize = z_RowSize(1, Sh)
Range(Cells(2, FromCol), Cells(RowSize, ToCol)).Select
'Rows("1:1").Find(What:="EtcTrialsFullCosts2010", LookAt:=xlWhole).Select
'Range(ActiveCell.End(xlToRight).Offset(1, -2), ActiveCell.End(xlDown)).Select 'offset?
'Range(ActiveCell.End(xlToRight), ActiveCell.End(xlDown)).Select
    Selection.NumberFormat = "#,##0"
    Selection.Replace What:=".", Replacement:=".", LookAt:=xlPart, _
        SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
        ReplaceFormat:=False
    Selection.ColumnWidth = 25
End Sub

```

Sub Run()

```
*****

'Open and activate an Excel workbook (and session)
Dim Wb As Workbook
Dim wbpath As String
Dim wbname As String
wbpath = "C:\Users\t740698\Desktop\"
wbname = "SmC_Download" & "_" & VBA.DateTime.Day(Now()) & "_" & _
    VBA.DateTime.Month(Now()) & "_" & VBA.DateTime.Year(Now()) & ".xlsb"
Call z_OpenAndActivateWb(wbname, wbpath, Wb)
*****

'move and resize excel session
Call z_ExcelSessionWindowNormal(Wb)
Call z_ExcelSessionWindowMoveAndResize(mywb, "0", "0", "700", "600")
*****

'Add and prepare the master dataset
Call z_ShNew("Log_1", "Begin")
Call z_ShNew("RD_MasterDataSet_1", "Begin")
Dim ShMaster_ColNames As Variant
ShMaster_ColNames = Array( _
    "PiIdentifier", "ActivityIdentifier", "SyngentaPortfolioLevel1", "SyngentaPortfolioLevel2",
    "PIPortfolioLevel3", "SyngentaPortfolio", "SyngentaProgram", "IsConfidential", "PiStatus",
    "PortfolioType", "PiSubType", "PiTitle", "PiManager", "PiSponsor", "PiResponsibility", "PiLabel",
    "PiStage", "LastGatePassed", "PiScope", "PiCustomer", "PiInvestmentCategory", "PiMarketSegment",
    "PiIndicatorSchedule", "PiIndicatorBudget", "PiIndicatorScope", "PiIndicatorQuality",
    "ReasonForDeviationScope", "ReasonForDeviationSchedule", "ReasonForDeviationQuality", _
    "ReasonForDeviationBudget", "PiLeadAi", "ListOfPiGrouping", "PiListOfActiveIngredients",
    "PiListOfCrops", "PiListOfCropsGroup", "PiListOfRegions", "PiListOfCountries", "PiGeography",
    "PiListOfProductFunctions", "PiPurchaseOrder", "PlanningItemName", "ActivityDescription",
    "ActivityType", "TaskTitle", "ListOfTaskCustomers", "Customer %", "TaskLocation", "TaskStatus",
    "WbsElement", "TaskContact", "ActivityComment", "LegacyTaskIdentifier", "Duration",
    "ExpectedFinishExport", "PlannedStart", "PlannedFinishExport", "ActualStart", "ActualFinishExport",
    "StartNoEarlierThan", _
    "FinishNoLaterThanExport", "AssignedResourcesWithLoad", "AssignedResourcesWithRate",
    "ResourceGroupDescription", "ResourceDescription", "RoleDescription", "TotalNpv", "SalesPeak",
    "BcRequired", "EtcTrialsFullCosts2010", "EtcTrialsFullCosts2011", "EtcTrialsFullCosts2012",
    "EtcTrialsFullCosts2013", "EtcTrialsFullCosts2014", "EtcTrialsFullCosts", "EtcSdFullCosts2010",
    "EtcSdFullCosts2011", "EtcSdFullCosts2012", "EtcSdFullCosts2013", "EtcSdFullCosts2014",
    "EtcSdFullCosts", "EtcOther2010", "EtcOther2011", "EtcOther2012", "EtcOther2013",
    "EtcOther2014", "EtcOther", "EtcFullCosts2010", "EtcFullCosts2011", "EtcFullCosts2012", _
    "EtcFullCosts2013", "EtcFullCosts2014", "EtcFullCosts", "EtcExt2010", "EtcExt2011",
    "EtcExt2012", "EtcExt2013", "EtcExt2014", "EtcExt", "EtcTrials2010", "EtcTrials2011", "EtcTrials2012",
    "EtcTrials2013", "EtcTrials2014", "EtcTrials", "EtcSd2010", "EtcSd2011", "EtcSd2012", "EtcSd2013",
    "EtcSd2014", "EtcSd", "EacTrialsFullCosts2010", "EacTrialsFullCosts2011", "EacTrialsFullCosts2012",
    "EacTrialsFullCosts2013", "EacTrialsFullCosts2014", "EacTrialsFullCosts", "EacSdFullCosts2010",
    "EacSdFullCosts2011", "EacSdFullCosts2012", _
    "EacSdFullCosts2013", "EacSdFullCosts2014", "EacSdFullCosts", "EacOther2010",
    "EacOther2011", "EacOther2012", "EacOther2013", "EacOther2014", "EacOther",
    "EacFullCosts2010", "EacFullCosts2011", "EacFullCosts2012", "EacFullCosts2013",
    "EacFullCosts2014", "EacFullCosts", "EacExt2010", "EacExt2011", "EacExt2012", "EacExt2013",
    "EacExt2014", "EacExt", "EacTrials2010", "EacTrials2011", "EacTrials2012", "EacTrials2013",
    "EacTrials2014", "EacTrials", "EacSd2010", "EacSd2011", "EacSd2012", _
```

```

"EacSd2013", "EacSd2014", "EacSd", "AcTrialsFullCosts2010", "AcTrialsFullCosts2011",
"AcTrialsFullCosts2012", "AcTrialsFullCosts2013", "AcTrialsFullCosts2014", "AcTrialsFullCosts",
"AcSdFullCosts2010", "AcSdFullCosts2011", "AcSdFullCosts2012", "AcSdFullCosts2013",
"AcSdFullCosts2014", "AcSdFullCosts", "AcOther2010", "AcOther2011", "AcOther2012",
"AcOther2013", "AcOther2014", "AcOther", "AcFullCosts2010", "AcFullCosts2011",
"AcFullCosts2012", "AcFullCosts2013", "AcFullCosts2014", "AcFullCosts", "AcExt2010", "AcExt2011",
"AcExt2012", _
"AcExt2013", "AcExt2014", "AcExt", "AcTrials2010", "AcTrials2011", "AcTrials2012",
"AcTrials2013", "AcTrials2014", "AcTrials", "AcSd2010", "AcSd2011", "AcSd2012", "AcSd2013",
"AcSd2014", "AcSd")
Call z_AddColNames(ShMaster_ColNames, "RD_MasterDataSet1", 1, Wb)
*****

'loop over all worksheets
Dim Sh As Worksheet
For Each Sh In Wb.Sheets
    'read the data into the master dataset
    z_FillTheMasterDataset (Wb)
Next Sh
*****

```

End Sub

Function z\_FillTheMasterDataset(Wb As Worksheet)

```

*****

'go through Sh_source col E (Pild, ActId, ResId)
'(if (left(col(E),2))= PI then) write Pild into Sh_target col A
'(if (left(col(E),2))= PI,WS,TK,MS) write ActId into Sh_target col B
'(if (left(col(E),2))= empty) write Pild&ActId&ResId into Sh_target col C
Call z_AddPildAndActivityIdAndRessourceId(Sh_source, Sh_target, Col_source, Col_target)
Call z_FillEmptyCellWithCellValueAbove(Sh_target, Col_target)
'go through Sh_source col E
'(if Pild_source=Pild_target then)
Call z_MapCols_Pild
*****

'go through Sh_source col E
'(if ActId_source=ActId_target then)
Call z_MapCols_ActId
*****

'go through Sh_source col E
'(if ActId_source=ActId_target then)
Call z_MapCols_ActId
*****

,

Dim ShPisEAC_ColNamesToMap As Variant
ShPisEAC_ColNamesToMap = Array( _
    "PIPortfolioLevel3", "SyngentaProgram", "PiStatus", "PortfolioType", "PiSubType", "PiTitle",
"PiManager", "PiSponsor", "PiResponsibility", "PiLabel", _
    "PiStage", "LastGatePassed", "PiScope", "PiCustomer", "PiInvestmentCategory",
"PiMarketSegment", "PiIndicatorSchedule", "PiIndicatorBudget", "PiIndicatorScope",
"PiIndicatorQuality", _
    "ReasonForDeviationScope", "ReasonForDeviationSchedule", "ReasonForDeviationQuality",
"ReasonForDeviationBudget", "PiLeadAi", _

```

```

    "ListOfPiGrouping", "PiListOfActiveIngredients", "PiListOfCrops", "PiListOfCropsGroup",
    "PiListOfRegions", "PiListOfCountries", "PiGeography", "PiListOfProductFunctions",
    "PiPurchaseOrder", _
    "TotalNpv", "SalesPeak", "BcRequired")
    Call z_ShMapColumns(ShPisEAC_ColNamesToMap, ShMaster_ColNames, "PisEAC_FV",
    "RD_MasterDataSet3", "Log_MapColPisEAC")
    *****
    ,
    Call z_SortAndFormat
    *****

```

End Function

```

Function z_ShMapColumns(ByRef Arr_ColNames As Variant, ByRef Arr_ColNames_new As Variant, _
    Sh As String, Sh_new As String, Sh_log As String, _
    Sh_KeyCol As String, Sh_KeyCol_new As String, _
    Optional ByRef Wb As Workbook)

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Application.ScreenUpdating = False

Dim Start As Date: Dim Duration As Date

Start = Now() 'to measure the duration of the code

Dim ilog As Integer

ilog = 2

'Determine the column indizes in Sh and Sh\_new,

Dim ColName\_i As String

ReDim ColIndices(0 To UBound(Arr\_ColNames), 0 To 2) As Variant

For i = LBound(Arr\_ColNames) To UBound(Arr\_ColNames) Step 1

ColName\_i = CStr(Arr\_ColNames(i))

ColIndices(i, 0) = ColName\_i

ColIndices(i, 1) = z\_GetColumnIndex(ColName\_i, 1, Sh)

ColIndices(i, 2) = z\_GetColumnIndex(ColName\_i, 1, Sh\_new)

Debug.Print ColIndices(i, 0)

Debug.Print ColIndices(i, 1)

Debug.Print ColIndices(i, 2)

If ColIndices(i, 1) = 0 Then

    'write errors into the logfile

    Sheets(Sh\_log).Cells(ilog, 2) = "ColNames"

    Sheets(Sh\_log).Cells(ilog, 3) = i

    Sheets(Sh\_log).Cells(ilog, 4) = "not found in Sh"

    ilog = ilog + 1

    Stop 'make sure you find all!!!!!!

Elseif ColIndices(i, 2) = 0 Then

    'write errors into the logfile

    Sheets(Sh\_log).Cells(ilog, 2) = "ColNames"

    Sheets(Sh\_log).Cells(ilog, 3) = i

    Sheets(Sh\_log).Cells(ilog, 4) = "not found in Sh\_new"

    ilog = ilog + 1

    Stop 'make sure you find all!!!!!!

End If

Next i

'Determine the row size in Sh\_new

```

Sheets(Sh_new).Activate
RowSize_new = z_RowSize(1, Sh_new)

'Check whether column F is the "PIdentifier" in "Sh"
If z_PIdentifierCheck(Sh, 1, 6) = False Then
    Stop
End If
'Check whether column A is the "PIdentifier" and select Column A "PIdentifier" in "Sh_new"
If z_PIdentifierCheck(Sh_new, 1, 1) = False Then
    Stop
Else
    Range(Cells(2, 1), Cells(RowSize_new, 1)).Select
End If

'Iterate through the rows with "rcheck" = PIdentifier
For Each rcheck In Selection.Cells
    'if rcheck is found in Sh then perform the mapping
    If Not Sheets(Sh).Columns("F:F").Find(What:=rcheck, LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns
        For j = LBound(ColIndices) To UBound(ColIndices) Step 1
            'read the indices
            ColIndex_j = ColIndices(j, 1)
            ColIndex_new_j = ColIndices(j, 2)
            'map
            rcheck.Offset(0, (ColIndex_new_j) - 1).Value = _
                Sheets(Sh).Columns("F:F").Find(What:=rcheck, LookAt:=xlWhole).Offset(0, (ColIndex_j) -
6)
        Next j
    Else
        'write errors into the logfile
        Sheets(Sh_log).Cells(iLog, 2) = rcheck 'writes out PIdentifier
        Sheets(Sh_log).Cells(iLog, 4) = "not found"
    End If
Next
Sheets(Sh_new).Rows.RowHeight = 15
'Write the durations into the logfile
Duration = Now() - Start
Sheets(Sh_log).Cells(iLog + 2, 1) = "Duration" & CStr(Duration): iLog = iLog + 1

Application.ScreenUpdating = True
Exit Function
NameExpectedNotExist:
    'write into the RD_MasterDataSet_1
    Sheets("RD_MasterDataSet_1").Cells(1, EnumColIndexExp_i) = ColNameExp_i
    Sheets("RD_MasterDataSet_1").Cells(1, EnumColIndexExp_i).Font.Color = RGB(255, 255, 0)
    'write into the logfile
    iLog = iLog + 1
    Sheets("Logfile_MapPIsEAC").Cells(iLog, 2) = ColNameExp_i
    Sheets("Logfile_MapPIsEAC").Cells(iLog, 3) = "not found": iLog = iLog + 1
    NotfoundFlag = 0
    Resume Next
End Function

```

```

Function z_PIdentifierCheck(Sh As String, Sh_row As Long, Sh_col As Long) As Boolean
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
If Sheets(Sh).Cells(Sh_row, Sh_col) = "PIdentifier" Then
    z_PIdentifierCheck = True
Else
    z_PIdentifierCheck = False
End If
End Function

```

```

Sub Run()
    *****

    'Open and activate an Excel workbook (and session)
    Dim Wb_1 As Workbook
    Dim wbpath As String
    Dim wbname As String
    wbpath = "C:\Users\t740698\Desktop\"
    wbname = "SmC_Download" & "_" & VBA.DateTime.Day(Now()) & "_" & _
        VBA.DateTime.Month(Now()) & "_" & VBA.DateTime.Year(Now()) & ".xlsb"
    'wbname = "SmC_Download_25_10_2011.xlsb" 'for test purposes
    Call z_OpenAndActivateWb(wbname, wbpath, Wb_1)
    *****

    'move and resize the Excel session
    Call z_ExcelSessionWindowNormal(Wb_1)
    Call z_ExcelSessionWindowMoveAndResize(Wb_1, "0", "0", "700", "600")
    *****

    'flattening of the Pi level worksheets Smc_P
    If 0 Then
        Dim Sh_from As Worksheet
        Dim Sh_to As String
        'loop over all worksheets
        For Each Sh_from In Wb_1.Sheets
            If Sh_from.Name <> "RD_MasterDataSet_1" And Sh_from.Name <> "Log_1" And Sh_from.Name
            <> "Logfile" Then
                If left(Sh_from.Name, 5) = "SmC_P" Then
                    Sh_to = Sh_from.Name & "_c"
                    Call z_SmC_CleanProjectReportPortfolioStructure(Sh_from.Name, Sh_to) '!!!!!!!!!!!!!!!!!!!!!!
                write the function
            End If
        End If
        Next Sh_from
    End If
    *****

    'loop over all worksheets and copy the content of the activity level SmC_A sheets
    'into the MasterDataSet
    If 0 Then
        Call z_ShNew("Log_1", "Begin")
        Call z_ShNew("RD_MasterDataSet_1", "Begin")
        Call z_ShNew("RD_PiDataSet_1", "Begin")
        For Each Sh_from In Wb_1.Sheets
            'write the data from Sh_from into the MasterDataSet
            If Sh_from.Name <> "RD_MasterDataSet_1" And Sh_from.Name <> "Log_1" And Sh_from.Name
            <> "Logfile" Then

```

```

        If left(Sh_from.Name, 5) = "SmC_A" Then
            Call z_CopySh1ToSh2_GivenColLastRow(Sh_from.Name, "Activity Identifier",
"RD_MasterDataSet_1") '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!testen-ok
        End If
    End If
    Next Sh_from
End If
'loop over all worksheets and copy the content of the Pi level SmC_P sheets
'into the MasterDataSet
If 0 Then
    For Each Sh_from In Wb_1.Sheets
        'write the data from Sh_from into the MasterDataSet
        If Sh_from.Name <> "RD_MasterDataSet_1" And Sh_from.Name <> "Log_1" And Sh_from.Name
<> "Logfile" Then
            If Right(Sh_from.Name, 2) = "_c" Then
                Call z_CopySh1ToSh2_GivenColLastRow(Sh_from.Name, "Portfolio Level 1",
"RD_PiDataSet_1") '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!testen-ok
            End If
        End If
    Next Sh_from
End If

'*****
'*****

'Add the new workbook RD_MasterDataSet
'*****
'*****

'Add a new workbook and move the MasterDataSet and the PiDataSet into it
Dim Wb_2 As Workbook
wbpath = "C:\Users\t740698\Desktop\"
wbname = "RD_MasterDataSet" & "_" & VBA.DateTime.Day(Now()) & "_" & _
VBA.DateTime.Month(Now()) & "_" & VBA.DateTime.Year(Now()) & ".xlsb"
'wbname = "RD_MasterDataSet_27_10_2011.xlsb" 'for test purposes
Call z_WorkbookNewOrOpenOrActivate(wbname, wbpath, Wb_2)
If 0 Then
    Call z_MoveSheetFromWb1ToWb2("Log_1", Wb_1, Wb_2, "Begin") '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Testen-ok
    Call z_MoveSheetFromWb1ToWb2("RD_PiDataSet_1", Wb_1, Wb_2, "Begin") '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Testen-ok
    Call z_MoveSheetFromWb1ToWb2("RD_MasterDataSet_1", Wb_1, Wb_2, "Begin")
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Testen-ok
    'in case they have already been deleted in a previous run
    On Error Resume Next
        Call z_ShDelete("Sheet1", Wb_2)
        Call z_ShDelete("Sheet2", Wb_2)
        Call z_ShDelete("Sheet3", Wb_2)
    On Error GoTo 0
    Call z_WorkbookSave(Wb_2) '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Testen-ok
End If

'*****

'move and resize the Excel session
Call z_ExcelSessionWindowNormal(Wb_2)
Call z_ExcelSessionWindowMoveAndResize(Wb_2, "0", "0", "900", "700")
'*****
'*****

```



```

'Processing of the RD_MasterDataSet
*****

'remove the first empty row and the column names in between the PiDataSet
If 0 Then
    Call z_ShNew("Log_2", "Begin")
    Call z_ShNewFlatValueCopy("RD_PiDataSet_1", "RD_PiDataSet_2", "Begin")
    Call z_DeleteRows("RD_PiDataSet_2", 1, 1)
    Call z_RemoveColNames("RD_PiDataSet_2", "Log_2", 1, "Portfolio Level 1")
End If
*****

'Write to each row of the MasterDataSet the Ids (Pild, ActId, ResId, SortId)
'1.add col A, col B, col C and col D
'2.go through Col_from (Pild, ActId, ResId)
'2.1(if (left(Col_from,2))= PI then) write Pild into col A
'2.2(if (left(Col_from,2))= PI,WS,TK,MS) write ActId into col B
'2.3(if (left(Col_from,2))= empty) write MyResId into col C
'2.4(if (left(Col_from,2))= PI then) write MySortId=1..PiRows.Count into col D
If 0 Then
    Call z_ShNew("Log_3", "Begin")
    Call z_ShNewFlatValueCopy("RD_MasterDataSet_1", "RD_MasterDataSet_2", "Begin")
    Call z_InsertEmptyCols("RD_MasterDataSet_2", 4, 1)
    Call z_DeleteRows("RD_MasterDataSet_2", 1, 1)
    Call z_AddColNames(Array("Pildentifier", "ActivityIdentifier", "MyRessourceId", "MySortId"),
"RD_MasterDataSet_2", 1)
    Call z_RemoveColNames("RD_MasterDataSet_2", "Log_3", 5, "Activity Identifier")
    Call z_AddWBSPild("RD_MasterDataSet_2", "Activity Identifier", "Pildentifier") '!!!!!!!!!!!!!!!!!!!!!!
ok
    Call z_AddWBSActivityId("RD_MasterDataSet_2", "Activity Identifier", "ActivityIdentifier")
'!!!!!!!!!!!!!!!!!!!!!!ok
    Call z_AddMyRessourceId("RD_MasterDataSet_2", "Activity Identifier", "MyRessourceId")
'!!!!!!!!!!!!!!!!!!!!!!ok
    Call z_AddMySortId("RD_MasterDataSet_2", "ActivityIdentifier", "MySortId", "WithoutMS")
'!!!!!!!!!!!!!!!!!!!!!!ok
    Call z_WorkbookSave(Wb_2)
End If
*****

'Copy the values of the summary lines into the lines below
If 0 Then
    Call z_ShNew("Log_4", "Begin")
    Call z_ShNewFlatValueCopy("RD_MasterDataSet_2", "RD_MasterDataSet_3", "Begin")
    Dim ColName_Array() As Variant
    Dim ColName As String
    Dim Col As Long
    ColName_Array = Array("Description", "Activity Comment", "Activity type", _
        "List of Task Customers", "Task Contact", "Task Location", _
        "Task Status", "Duration") '!!!!!!!!!!!!!!!!!!!!!!ok
    For Iter = LBound(ColName_Array) To UBound(ColName_Array)
        ColName = ColName_Array(Iter)
        Call z_CopyWBSAttributeEntriesIntoRessources("RD_MasterDataSet_3", "ActivityIdentifier",
ColName) '!!!!!!!!!!!!!!!!!!!!!!ok
    Next Iter
    ColName = "Syngenta Portfolio"

```

```

Call z_CopyWBSAttributeEntriesIntoResources("RD_MasterDataSet_3", "PIdentifier", ColName)
Call z_WorkbookSave(Wb_2)
End If
*****

'Add empty columns an write out the column name
Dim FirstYr As Integer
FirstYr = 2010
Dim Col_to As Long
If 0 Then
Call z_ShNew("Log_5", "Begin")
Call z_ShNewFlatValueCopy("RD_MasterDataSet_3", "RD_MasterDataSet_4", "Begin")
'add EAC Full Cost columns
Col_to = z_GetColumnIndex("ETC Full Costs", 1, "RD_MasterDataSet_4") + 1
Call z_InsertEmptyCols("RD_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Full Costs " & CStr(FirstYr) & "_c", "EAC Full Costs " & CStr(FirstYr +
1) & "_c", _
"EAC Full Costs " & CStr(FirstYr + 2) & "_c", "EAC Full Costs " & CStr(FirstYr + 3) & "_c", _
"EAC Full Costs " & CStr(FirstYr + 4) & "_c", "EAC Full Costs " & "_c"), "RD_MasterDataSet_4",
1, Col_to)
'add EAC SD Full Costs columns
Col_to = z_GetColumnIndex("EAC Full Costs " & "_c", 1, "RD_MasterDataSet_4") + 1
Call z_InsertEmptyCols("RD_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC SD Full Costs " & CStr(FirstYr) & "_c", "EAC SD Full Costs " &
CStr(FirstYr + 1) & "_c", _
"EAC SD Full Costs " & CStr(FirstYr + 2) & "_c", "EAC SD Full Costs " & CStr(FirstYr + 3) & "_c",
_
"EAC SD Full Costs " & CStr(FirstYr + 4) & "_c", "EAC SD Full Costs " & "_c"),
"RD_MasterDataSet_4", 1, Col_to)
'add EAC Trials Full Cost columns
Col_to = z_GetColumnIndex("EAC SD Full Costs " & "_c", 1, "RD_MasterDataSet_4") + 1
Call z_InsertEmptyCols("RD_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Trials Full Costs " & CStr(FirstYr) & "_c", "EAC Trials Full Costs " &
CStr(FirstYr + 1) & "_c", _
"EAC Trials Full Costs " & CStr(FirstYr + 2) & "_c", "EAC Trials Full Costs " & CStr(FirstYr + 3) &
_c", _
"EAC Trials Full Costs " & CStr(FirstYr + 4) & "_c", "EAC Trials Full Costs " & "_c"),
"RD_MasterDataSet_4", 1, Col_to)
'add EACOther$ columns
Col_to = z_GetColumnIndex("EAC Trials Full Costs " & "_c", 1, "RD_MasterDataSet_4") + 1
Call z_InsertEmptyCols("RD_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Other $ " & CStr(FirstYr) & "_c", "EAC Other $ " & CStr(FirstYr + 1)
& "_c", _
"EAC Other $ " & CStr(FirstYr + 2) & "_c", "EAC Other $ " & CStr(FirstYr + 3) & "_c", _
"EAC Other $ " & CStr(FirstYr + 4) & "_c", "EAC Other $" & "_c"), "RD_MasterDataSet_4", 1,
Col_to)
'add EACExt$ columns
Col_to = z_GetColumnIndex("EAC Other $" & "_c", 1, "RD_MasterDataSet_4") + 1
Call z_InsertEmptyCols("RD_MasterDataSet_4", 6, Col_to)
Call z_AddColNames(Array("EAC Ext $ " & CStr(FirstYr) & "_c", "EAC Ext $ " & CStr(FirstYr + 1) &
_c", _
"EAC Ext $ " & CStr(FirstYr + 2) & "_c", "EAC Ext $ " & CStr(FirstYr + 3) & "_c", _
"EAC Ext $ " & CStr(FirstYr + 4) & "_c", "EAC Ext $" & "_c"), "RD_MasterDataSet_4", 1,
Col_to)

```

```

End If
*****

'sort the dataset using col 3 to keep the ressources (or col 4 to only remove the MS but keep the
summary lines)
If 0 Then
    Call z_ShNew("Log_6", "Begin")
    Call z_ShNewFlatValueCopy("RD_MasterDataSet_4", "RD_MasterDataSet_5", "Begin")
    Call z_Sort("RD_MasterDataSet_5", 3, 3) '!!!!!!!!!!!!!!!!!!!!Testen-ok
    Call z_DeleteNotRessourceRows("RD_MasterDataSet_5", "MyRessourceId")
    Call z_WorkbookSave(Wb_2)
End If
*****

'start the cost calculation
If 0 Then
    Call z_ShNew("Log_7", "Begin")
    Call z_ShNewFlatValueCopy("RD_MasterDataSet_5", "RD_MasterDataSet_6", "Begin")
    Dim FromCol As Long
    Dim ToCol As Long
    FromCol = z_GetColumnIndex("EAC SD Full Costs " & CStr(FirstYr), 1, "RD_MasterDataSet_6")
    ToCol = z_GetColumnIndex("EAC Ext $" & "_c", 1, "RD_MasterDataSet_6")
    Call z_ChgFmt_CostCols("RD_MasterDataSet_6", FromCol, ToCol)
    Call CostCalc("RD_MasterDataSet_6", "PiIdentifier", FirstYr, "AC Full Costs", "ETC Full Costs",
"Unit", _
                "EAC SD Full Costs", "EAC Trials Full Costs", "EAC Other $", "EAC Ext $", "EAC Full Costs",
"Log_7")
    Call z_WorkbookSave(Wb_2)
End If
*****

'Map colums form Sh_source to Sh_target by comparing the key
If 0 Then
    Call z_ShNew("Log_8", "Begin")
    Call z_ShNewFlatValueCopy("RD_MasterDataSet_6", "RD_MasterDataSet_7", "Begin")

    'Define the attribute names to map into the Sh_to
    Dim ColNames_to As Variant
    ColNames_to = Array( _
        "PIPortfolioLevel3", "SyngentaProgram", "PiStatus", "PortfolioType", "PiSubType", "PiTitle",
"PiManager", "PiSponsor", "PiResponsibility", "PiLabel", _
        "PiStage", "LastGatePassed", "PiScope", "PiCustomer", "PiInvestmentCategory",
"PiMarketSegment", "PiIndicatorSchedule", "PiIndicatorBudget", "PiIndicatorScope",
"PiIndicatorQuality", _
        "ReasonForDeviationScope", "ReasonForDeviationSchedule", "ReasonForDeviationQuality",
"ReasonForDeviationBudget", "PiLeadAi", _
        "ListOfPiGrouping", "PiListOfActiveIngredients", "PiListOfCrops", "PiListOfCropsGroup",
"PiListOfRegions", "PiListOfCountries", "PiGeography", "PiListOfProductFunctions",
"PiPurchaseOrder", _
        "TotalNpv", "SalesPeak", "BcRequired") '!!!!!!!!!!!!!!!!!!!!Add the col names to map

    'Write the attribute names at the first empty column in Sh_to
    Dim ColStart As Long
    ColStart = z_ColSize(1, "RD_MasterDataSet_7") + 1
    Call z_AddColNames(ColNamesToMap, "RD_MasterDataSet_7", 1, ColStart)

```

```

'Define the attribute names in Sh_from
'Even though the attribute names in Sh_from and Sh_to may be called differently they must
'refer to the same attribute and the have to be in the same order in both arrays!!!!
Dim ColNames_from As Variant
ColNames_from = ColNames_to

'check whether the column names exist in Sh_from
Sh_from = "PisEAC"
'Call z_ColNamesExist(ColNames_from, Sh_from) '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!look into shmapcolumns()

'call the map function
'Call z_ShMapColumns(Sh_from, "Pidentifier", ColNames_from, "RD_MasterDataSet_5",
"Pidentifier", ColNames_to, "Log_5")
Call z_WorkbookSave(Wb_2)
End If
'*****

'copy only the needed cols into a new sheet
'the array determines the order of the columns
If 0 Then
Call z_ShNew("Log_7", "Begin")
Call z_ShNewFlatValueCopy("RD_MasterDataSet_5", "RD_MasterDataSet_6", "Begin")
'create the ColName_Arr with the function z_CreateArrayInput_AttributeNames
'Call z_CreateArrayInput_AttributeNames
Dim ColName_Arr As Variant
ColName_Arr = Array("Pidentifier", "ActivityIdentifier", "SyngentaPortfolioLevel1",
"SyngentaPortfolioLevel2", "PiPortfolioLevel3", "SyngentaPortfolio", "SyngentaProgram",
"IsConfidential", "PiStatus", "PortfolioType", "PiSubType", "PiTitle", "PiManager", "PiSponsor",
"PiResponsibility", "PiLabel", "PiStage", "LastGatePassed", "PiScope", "PiCustomer",
"PiInvestmentCategory", "PiMarketSegment", "PiIndicatorSchedule", "PiIndicatorBudget",
"PiIndicatorScope", "PiIndicatorQuality", "ReasonForDeviationScope",
"ReasonForDeviationSchedule", "ReasonForDeviationQuality", _
"ReasonForDeviationBudget", "PiLeadAi", "ListOfPiGrouping", "PiListOfActiveIngredients",
"PiListOfCrops", "PiListOfCropsGroup", "PiListOfRegions", "PiListOfCountries", "PiGeography",
"PiListOfProductFunctions", "PiPurchaseOrder", "PlanningItemName", "ActivityDescription",
"ActivityType", "TaskTitle", "ListOfTaskCustomers", "Customer %", "TaskLocation", "TaskStatus",
"WbsElement", "TaskContact", "ActivityComment", "LegacyTaskIdentifier", "Duration",
"ExpectedFinishExport", "PlannedStart", "PlannedFinishExport", "ActualStart", "ActualFinishExport",
"StartNoEarlierThan", _
"FinishNoLaterThanExport", "AssignedResourcesWithLoad", "AssignedResourcesWithRate",
"ResourceGroupDescription", "ResourceDescription", "RoleDescription", "TotalNpv", "SalesPeak",
"BcRequired", "EtcTrialsFullCosts2010", "EtcTrialsFullCosts2011", "EtcTrialsFullCosts2012",
"EtcTrialsFullCosts2013", "EtcTrialsFullCosts2014", "EtcTrialsFullCosts", "EtcSdFullCosts2010",
"EtcSdFullCosts2011", "EtcSdFullCosts2012", "EtcSdFullCosts2013", "EtcSdFullCosts2014",
"EtcSdFullCosts", "EtcOther2010", "EtcOther2011", "EtcOther2012", "EtcOther2013",
"EtcOther2014", "EtcOther", "EtcFullCosts2010", "EtcFullCosts2011", "EtcFullCosts2012", _
"EtcFullCosts2013", "EtcFullCosts2014", "EtcFullCosts", "EtcExt2010", "EtcExt2011",
"EtcExt2012", "EtcExt2013", "EtcExt2014", "EtcExt", "EtcTrials2010", "EtcTrials2011", "EtcTrials2012",
"EtcTrials2013", "EtcTrials2014", "EtcTrials", "EtcSd2010", "EtcSd2011", "EtcSd2012", "EtcSd2013",
"EtcSd2014", "EtcSd", "EacTrialsFullCosts2010", "EacTrialsFullCosts2011", "EacTrialsFullCosts2012",
"EacTrialsFullCosts2013", "EacTrialsFullCosts2014", "EacTrialsFullCosts", "EacSdFullCosts2010",
"EacSdFullCosts2011", "EacSdFullCosts2012", _
"EacSdFullCosts2013", "EacSdFullCosts2014", "EacSdFullCosts", "EacOther2010",
"EacOther2011", "EacOther2012", "EacOther2013", "EacOther2014", "EacOther",

```

```

"EacFullCosts2010", "EacFullCosts2011", "EacFullCosts2012", "EacFullCosts2013",
"EacFullCosts2014", "EacFullCosts", "EacExt2010", "EacExt2011", "EacExt2012", "EacExt2013",
"EacExt2014", "EacExt", "EacTrials2010", "EacTrials2011", "EacTrials2012", "EacTrials2013",
"EacTrials2014", "EacTrials", "EacSd2010", "EacSd2011", "EacSd2012", _
    "EacSd2013", "EacSd2014", "EacSd", "AcTrialsFullCosts2010", "AcTrialsFullCosts2011",
"AcTrialsFullCosts2012", "AcTrialsFullCosts2013", "AcTrialsFullCosts2014", "AcTrialsFullCosts",
"AcSdFullCosts2010", "AcSdFullCosts2011", "AcSdFullCosts2012", "AcSdFullCosts2013",
"AcSdFullCosts2014", "AcSdFullCosts", "AcOther2010", "AcOther2011", "AcOther2012",
"AcOther2013", "AcOther2014", "AcOther", "AcFullCosts2010", "AcFullCosts2011",
"AcFullCosts2012", "AcFullCosts2013", "AcFullCosts2014", "AcFullCosts", "AcExt2010", "AcExt2011",
"AcExt2012", _
    "AcExt2013", "AcExt2014", "AcExt", "AcTrials2010", "AcTrials2011", "AcTrials2012",
"AcTrials2013", "AcTrials2014", "AcTrials", "AcSd2010", "AcSd2011", "AcSd2012", "AcSd2013",
"AcSd2014", "AcSd") '!!!!!!!!!!!!!!!!!!!!!!!!!!!!All Names in the wanted order
For Iter = LBound(ColName_Arr) To LBound(ColName_Arr)
    Dim Col_to_i As Long
    Dim Col_from_i As Long
    Col_to_i = Iter + 1
    ColName_i = ColName_Arr(Iter)
    'Col_from_i = z_GetColumnIndex(ColName_i, 1, "RD_MasterDataSet_5")
    Call z_CopyColumn("RD_MasterDataSet_5", Col_from_i, "RD_MasterDataSet_6", Col_to_i)
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!Testen-ok
    Next Iter
    Call z_WorkbookSave(Wb_2)
End If

'*****

'Give each col the right format
Call z_Format
'*****

'Delete the summary rows
If 0 Then
    Call z_ShNew("Log_8", "Begin")
    Call z_ShNewFlatValueCopy("RD_MasterDataSet_6", "RD_MasterDataSet_7", "Begin")
    Dim SearchCol As Long
    SearchCol = z_GetColumnIndex("IdentifierColName???", 1, "RD_MasterDataSet_7")
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!Add Col Name
    Call z_DeleteRowsOfNotEmptyCells("RD_MasterDataSet_7", SearchCol, 2) '!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Testen-ok
End If

'*****

'Create a pivot table

'*****

'Create the exception reports, which are then used to improve data quality in SmC

'*****

End Sub

Function CostCalc(Sh As String, Pild As String, YearStart As Integer, AcFull As String, EtcFull As String,
Unit As String, _
    EacSD As String, EacTrial As String, EacOther As String, EacExt As String, EacFull As String, _
    Sh_log As String)
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

Application.ScreenUpdating = False
Dim Pild_Col As Long
Dim Unit_Col As Long
Dim AcFull_Col As Long
Dim EtcFull_Col As Long
Dim EacFull_Col As Long
Dim EacSD_Col As Long
Dim EacTrial_Col As Long
Dim EacOther_Col As Long
Dim EacExt_Col As Long

'get the column index of the unit attribute and the Pild attribute
Unit_Col = z_GetColumnIndex(Unit, 1, Sh)
Pild_Col = z_GetColumnIndex(Pild, 1, Sh)

'iterate through all rows
Dim Search_Col As Long
Search_Col = z_GetColumnIndex(Pild, 1, Sh)
RowSize = z_RowSize(Search_Col, Sh)
For Row = 2 To RowSize
    If (Row Mod 100) = 0 Then
        Debug.Print Row
    End If
    'iterate through all Years
    Dim YearEnd As Integer
    YearEnd = YearStart + 4
    For Yr_iter = YearStart To YearEnd + 1
        'For the costs in a year
        If Yr_iter < YearEnd Or Yr_iter = YearEnd Then
            Yr = " " & CStr(Yr_iter)
            'For the total costs over all years
        Else
            Yr = ""
        End If
        'Get the column indices of the cost attributes
        AcFull_Col = z_GetColumnIndex(AcFull & Yr, 1, Sh)
        EtcFull_Col = z_GetColumnIndex(EtcFull & Yr, 1, Sh)
        EacFull_Col = z_GetColumnIndex(EacFull & Yr & "_c", 1, Sh)
        EacSD_Col = z_GetColumnIndex(EacSD & Yr & "_c", 1, Sh)
        EacTrial_Col = z_GetColumnIndex(EacTrial & Yr & "_c", 1, Sh)
        EacOther_Col = z_GetColumnIndex(EacOther & Yr & "_c", 1, Sh)
        EacExt_Col = z_GetColumnIndex(EacExt & Yr & "_c", 1, Sh)
        'Write out each time a calculation is necessary
        Dim Row_log As Long
        Row_log = 2
        If Cells(Row, AcFull_Col) <> Empty Then
            Sheets(Sh_log).Cells(Row_log, 1) = Cells(Row, Pild_Col)
            Sheets(Sh_log).Cells(Row_log, 1) = Cells(Row, AcFull_Col)
            Row_log = Row_log + 1
        End If
        'Fill the cost cell with the calculated cost
        'Either add the SD Costs
        If left(Cells(Row, Unit_Col), 4) = "SD__" Or left(Cells(Row, Unit_Col), 2) = "SD" Then

```

```

        Cells(Row, EacSD_Col) = Cells(Row, AcFull_Col) + Cells(Row, EtcFull_Col)
        Cells(Row, EacFull_Col) = Cells(Row, EacSD_Col)
    'or add the Trial Costs
    ElseIf left(Cells(Row, Unit_Col), 4) = "TRIA" Then
        Cells(Row, EacTrial_Col) = Cells(Row, AcFull_Col) + Cells(Row, EtcFull_Col)
        Cells(Row, EacFull_Col) = Cells(Row, EacTrial_Col)
    'or add the Other$ Costs
    ElseIf left(Cells(Row, Unit_Col), 4) = "$_OT" Then
        Cells(Row, EacOther_Col) = Cells(Row, AcFull_Col) + Cells(Row, EtcFull_Col)
        Cells(Row, EacFull_Col) = Cells(Row, EacOther_Col)
    'or add the External$ Costs
    ElseIf left(Cells(Row, Unit_Col), 4) = "$_EX" Then
        Cells(Row, EacExt_Col) = Cells(Row, AcFull_Col) + Cells(Row, EtcFull_Col)
        Cells(Row, EacFull_Col) = Cells(Row, EacExt_Col)
    'New unit to add to this else-if-tree
    Else
        Stop 'error: a new resource unit was added, add it to this else-if-tree
    End If
Next Yr_iter
Next Row
Application.ScreenUpdating = True
End Function

```

```

Function z_SmC_CleanProjectReportPortfolioStructure(Sh_from As String, Sh_to As String)
    'A user of this macro needs to provide an extract of the SmartChoice Project report.
    'The macro cleans the first three portfolio levels (4 levels if SYNGENTA is also counted).
    'Activate Sh_from
    Sheets(Sh_from).Activate
    Dim IntPBSValue As String, IntPBSValue2 As String
    'Generate a copy of the sheet before changing the content
    Sheets(Sh_from).Copy After:=Sheets(1)
    ActiveSheet.Name = Sh_to

    'Generally "unmerge" merged cells
    Cells.MergeCells = False
    'Remove the SYNGENTA level
    If Cells(2, 6).Value = "SYNGENTA" Then
        Cells(2, 6).EntireRow.Delete
    End If
    'Remove not used columns
    Columns("A:A").EntireColumn.Delete
    Columns("D:D").EntireColumn.Delete
    'Add headers for the columns
    Range("A:C").ColumnWidth = 20
    Cells(1, 1).Value = "Portfolio Level 1"
    Cells(1, 2).Value = "Portfolio Level 2"
    Cells(1, 3).Value = "Portfolio Level 3"
    'Assign PBS level 1, 2 and 3 values to the cells on the left of each entry
    'PBS level 1
    Cells(2, 4).Select
    Do While Not ActiveCell.Value = ""
        If Not ActiveCell(1, -2).Borders(xlEdgeRight).LineStyle = xlContinuous Then
            IntPBSValue = ActiveCell.Value

```

```

        ActiveCell.EntireRow.Delete
        Do While ActiveCell(1, -2).Borders(xlEdgeRight).LineStyle = xlContinuous
            ActiveCell(1, -2).Value = IntPBSValue
            ActiveCell(2, 1).Select
        Loop
    End If
Loop
'PBS level 2
Cells(2, 4).Select
Do While Not ActiveCell.Value = ""
    If Not ActiveCell(1, 2).Value = ActiveCell(1, -2).Value Then
        If Not ActiveCell(1, -1).Borders(xlEdgeRight).LineStyle = xlContinuous Then
            IntPBSValue = ActiveCell.Value
            ActiveCell.EntireRow.Delete
            Do While ActiveCell(1, -1).Borders(xlEdgeRight).LineStyle = xlContinuous
                ActiveCell(1, -1).Value = IntPBSValue
                ActiveCell(2, 1).Select
            Loop
        Else: ActiveCell(2, 1).Select
        End If
    Else: ActiveCell(2, 1).Select
    End If
Loop
'PBS level 3
Cells(2, 4).Select
Do While Not ActiveCell.Value = ""
    If Not ActiveCell.Value = ActiveCell(1, -1).Value Then
        If Not ActiveCell.Borders(xlEdgeRight).LineStyle = xlContinuous Then
            IntPBSValue = ActiveCell.Value
            ActiveCell.EntireRow.Delete
            Do While ActiveCell.Borders(xlEdgeRight).LineStyle = xlContinuous And ActiveCell(1,
0).Borders(xlEdgeLeft).LineStyle = xlContinuous And ActiveCell(1, 0).Borders(xlEdgeRight).LineStyle =
xlContinuous
                ActiveCell(1, 0).Value = IntPBSValue
                ActiveCell(2, 1).Select
            Loop
        Else: ActiveCell(2, 1).Select
        End If
    Else: ActiveCell(2, 1).Select
    End If
Loop
'Remove not used columns
Columns("D:D").EntireColumn.Delete
Cells(1, 1).Select
End Function

```

```

Sub test43()
    Call z_DeleteNotRessourceRows("RD_MasterDataSet_4", "MyRessourceId")
End Sub

```

```

Function z_DeleteNotRessourceRows(Sh As String, ColName As String)
    Sheets(Sh).Activate

```



```

Dim Col As Long
Col = z_GetColumnIndex(ColName, 1, Sh)
Dim RowSize1 As Long
RowSize1 = z_RowSize(Col, Sh) + 1
Dim ColSize As Long
ColSize = z_ColSize(1, Sh)
Dim RowSize2 As Long
RowSize2 = z_LastWrittenRow(Sh, ColSize)
Range(Cells(RowSize1, 1), Cells(RowSize2, ColSize)).Select
Selection.ClearContents
End Function

```

```

Function z_RemoveColNames(Sh As String, Sh_log As String, Col_Start As Long, Criteria As String)
'Activate Sh and the filter
Sheets(Sh).Select
RowSize = z_RowSize(5, Sh)
ColSize = z_ColSize(Col_Start, Sh)
If Sheets(Sh).AutoFilterMode = False Then
    Rows(1).AutoFilter 'Data>Filter
End If
'Filter the wrong entries
ActiveSheet.Range(Cells(2, Col_Start), Cells(RowSize, ColSize)).AutoFilter _
    Field:=Col_Start, Criteria1:=Criteria
'Write out into the logfile
Cells.Copy Destination:=Sheets(Sh_log).Cells(1, 1)
'delete rows
Range(Cells(2, Col_Start), Cells(RowSize, Col_Start)).Select
Selection.EntireRow.Delete
'Undo the filter
Sheets(Sh).AutoFilterMode = False
End Function

```

```

Function z_Format()

```

```

End Function

```

```

Function z_CopySh1ToSh2_GivenColLastRow(Sh_from As String, ColNameStart_from As String, Sh_to
As String)
    Dim ColStart_from As Long
    ColStart_from = z_GetColumnIndex(ColNameStart_from, 1, Sh_from)
    Dim ColSize_from As Long
    ColSize_from = z_ColSize(1, Sh_from)
    'RowSize_from = z_RowSize(1, Sh_from)
    RowSize_from = z_LastWrittenRow(Sh_from, ColSize_from)
    Sheets(Sh_from).Activate
    Range(Cells(1, ColStart_from), Cells(RowSize_from, ColSize_from)).Select
    Selection.Copy
    'RowSize_to = z_RowSize(1, Sh_to) + 1
    RowSize_to = z_LastWrittenRow(Sh_to, ColSize_from) + 1
    Sheets(Sh_to).Activate
    Sheets(Sh_to).Cells(RowSize_to, 1).Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _

```

```
:=False, Transpose:=False  
End Function
```

```
Function z_AddWBSPild(Sh As String, KeyName As String, AttributeName As String)
```

```
    Sheets(Sh).Activate  
    Dim Col_Key As Long  
    Dim Col_Attr As Long  
    Col_Key = z_GetColumnIndex(KeyName, 1, Sh)  
    Col_Attr = z_GetColumnIndex(AttributeName, 1, Sh)  
    Dim ColSize As Long  
    Dim RowSize As Long  
    ColSize = z_ColSize(1, Sh)  
    RowSize = z_LastWrittenRow(Sh, ColSize)  
    Dim Pild As String
```

```
    'loop through all rows  
    For Row = 1 To RowSize  
        If (Row Mod 1000) = 0 Then  
            Debug.Print Row  
        End If  
        'find a new Pi  
        'Cells(Row, Col_Key).Select  
        If left(Cells(Row, Col_Key).Value, 2) = "PI" Then  
            'store the value of the key column on the Pi level  
            Pild = Cells(Row, Col_Key).Value  
            'write the value to the attr column on the Pi level  
            Cells(Row, Col_Attr).Value = Pild  
            Row = Row + 1  
            'Cells(Row, Col_Key).Select  
            'go through all rows of a Pi and write that stored value  
            Do While Not left(Cells(Row, Col_Key).Value, 2) = "PI"  
                If Cells(Row, Col_Key).Value = "Activity Identifier" Then  
                    Row = Row + 1  
                Else  
                    Cells(Row, Col_Attr).Value = Pild  
                    Row = Row + 1  
                    'Cells(Row, Col_Key).Select  
                    'termination criteria  
                    If Row = RowSize + 1 Then  
                        Exit Function  
                    End If  
                End If  
            Loop  
            'Iterate down to remain at the same row after the next statement iteration  
            Row = Row - 1  
        End If  
    Next Row  
End Function
```

```
Function z_AddWBSActivityId(Sh As String, KeyName As String, AttributeName As String)
```

```
    Sheets(Sh).Activate  
    Dim Col_Key As Long  
    Dim Col_Attr As Long
```

```

Col_Key = z_GetColumnIndex(KeyName, 1, Sh)
Col_Attr = z_GetColumnIndex(AttributeName, 1, Sh)
Dim ColSize As Long
Dim RowSize As Long
ColSize = z_ColSize(1, Sh)
RowSize = z_LastWrittenRow(Sh, ColSize)
Dim Pild As String

'loop through all rows
Dim Row As Long
Row = 1
Do While Row < RowSize + 1
    If (Row Mod 1000) = 0 Then
        Debug.Print Row
    End If
    'find a new Pi
    'Cells(Row, Col_Key).Select
    If left(Cells(Row, Col_Key).Value, 2) = "PI" Then
        'store the value of the key column on the Pi level
        Pild = Cells(Row, Col_Key).Value
        'write the value to the attr column on the Pi level
        Cells(Row, Col_Attr).Value = Pild
        Row = Row + 1
        'Cells(Row, Col_Key).Select
        'go through all rows of a Pi and write that stored value
        Do While left(Cells(Row, Col_Key).Value, 2) = Empty
            Cells(Row, Col_Attr).Value = Pild
            Row = Row + 1
            'Cells(Row, Col_Key).Select
            'termination criteria
            If Row = RowSize + 1 Then
                Exit Function
            End If
        Loop
        'Cells(Row, Col_Key).Select
        Row = Row - 1
        'find a new WS
        ElseIf left(Cells(Row, Col_Key).Value, 2) = "WS" Then
            'store the value of the key column on the WS level
            WsID = Cells(Row, Col_Key).Value
            'write the value to the attr column on the WS level
            Cells(Row, Col_Attr).Value = WsID
            Row = Row + 1
            'Cells(Row, Col_Key).Select
            'go through all rows of a WS and write that stored value
            Do While left(Cells(Row, Col_Key).Value, 2) = Empty
                Cells(Row, Col_Attr).Value = WsID
                Row = Row + 1
                'Cells(Row, Col_Key).Select
                'termination criteria
                If Row = RowSize + 1 Then
                    Exit Function
                End If
            End If
        End If
    End If

```

```

    Loop
    'Cells(Row, Col_Key).Select
    Row = Row - 1
    'find a new TK
    ElseIf left(Cells(Row, Col_Key).Value, 2) = "TK" Then
        'store the value of the key column on the WS level
        TkID = Cells(Row, Col_Key).Value
        'write the value to the attr column on the WS level
        Cells(Row, Col_Attr).Value = TkID
        Row = Row + 1
        'Cells(Row, Col_Key).Select
        'go through all rows of a WS and write that stored value
        Do While left(Cells(Row, Col_Key).Value, 2) = Empty
            Cells(Row, Col_Attr).Value = TkID
            Row = Row + 1
            'Cells(Row, Col_Key).Select
            'termination criteria
            If Row = RowSize + 1 Then
                Exit Function
            End If
        Loop
    'Cells(Row, Col_Key).Select
    Row = Row - 1
    'find a new MS
    ElseIf left(Cells(Row, Col_Key).Value, 2) = "MS" Then
        Cells(Row, Col_Attr).Value = Cells(Row, Col_Key).Value
    Else
        'rows with Ressources or column names
    End If
    Row = Row + 1
Loop
End Function

```

```

Function z_AddMyRessourceId(Sh As String, KeyName As String, AttributeName As String)
    Sheets(Sh).Activate
    Dim Col_Key As Long
    Dim Col_Attr As Long
    Col_Key = z_GetColumnIndex(KeyName, 1, Sh)
    Col_Attr = z_GetColumnIndex(AttributeName, 1, Sh)
    Dim ColSize As Long
    Dim RowSize As Long
    ColSize = z_ColSize(1, Sh)
    RowSize = z_LastWrittenRow(Sh, ColSize)
    Dim Pild As String
    Dim Cnt As Long
    Cnt = 1
    For Row = 1 To RowSize
        If (Row Mod 1000) = 0 Then
            Debug.Print Row
        End If
        'find a new Ressource RS
        'Cells(Row, Col_Key).Select
        If left(Cells(Row, Col_Key).Value, 2) = Empty Then

```

```

        Cells(Row, Col_Attr).Value = "RS" & Right("00000000" & CStr(Cnt), 8)
        Cnt = Cnt + 1
    End If
Next Row
End Function
Sub test23()
Call z_Sort("RD_MasterDataSet_4", 3, 3)
End Sub

```

```

Function z_AddMySortId(Sh As String, KeyName As String, AttributeName As String, ExceptionFlag As String)

```

```

    Sheets(Sh).Activate
    Dim Col_Key As Long
    Dim Col_Attr As Long
    Col_Key = z_GetColumnIndex(KeyName, 1, Sh)
    Col_Attr = z_GetColumnIndex(AttributeName, 1, Sh)
    Dim ColSize As Long
    Dim RowSize As Long
    ColSize = z_ColSize(1, Sh)
    RowSize = z_LastWrittenRow(Sh, ColSize)
    Dim Pild As String
    Dim Cnt As Long

    'loop through all rows
    Cnt = 1
    For Row = 2 To RowSize
        If (Row Mod 1000) = 0 Then
            Debug.Print Row
        End If
        If ExceptionFlag = "WithoutMS" Then
            If left(Cells(Row, Col_Key).Value, 2) <> "MS" And Cells(Row, Col_Key).Value <> Empty Then
                Cells(Row, Col_Attr).Value = Cnt
                Cnt = Cnt + 1
            End If
        End If
        If ExceptionFlag = "WithoutAttributeNames" Then
            If left(Cells(Row, Col_Key).Value, 4) <> "PI I" And Cells(Row, Col_Key).Value <> Empty Then
                Cells(Row, Col_Attr).Value = Cnt
                Cnt = Cnt + 1
            End If
        End If
    Next Row
End Function

```

```

Function z_AddMySortId1(Sh As String, KeyName As String, AttributeName As String)

```

```

    Sheets(Sh).Activate
    Dim Col_Key As Long
    Dim Col_Attr As Long
    Col_Key = z_GetColumnIndex(KeyName, 1, Sh)
    Col_Attr = z_GetColumnIndex(AttributeName, 1, Sh)
    Dim ColSize As Long
    Dim RowSize As Long
    ColSize = z_ColSize(1, Sh)

```

```

RowSize = z_LastWrittenRow(Sh, ColSize)
Dim Pild As String
Dim Cnt As Long

'loop through all rows
For Row = 1 To RowSize
    If (Row Mod 1000) = 0 Then
        Debug.Print Row
    End If
    'find a new Pi
    'Cells(Row, Col_Key).Select
    Cnt = 1
    If Left(Cells(Row, Col_Key).Value, 2) = "PI" Then
        Key = Cells(Row, Col_Key).Value
        Cells(Row, Col_Attr).Value = Cnt
        Cnt = Cnt + 1
        Row = Row + 1
        'Cells(Row, Col_Key).Select
        'go through all rows of a Pi and count up
        Do While Key = Cells(Row, Col_Key).Value
            Cells(Row, Col_Attr).Value = Cnt
            Cnt = Cnt + 1
            Row = Row + 1
            'Cells(Row, Col_Key).Select
            'termination criteria
            If Row = RowSize + 1 Then
                Exit Function
            End If
        Loop
        'Iterate down to remain at the same row after the next statement iteration
        Row = Row - 1
    End If
Next Row
End Function

Function z_AddMySortId2(Sh As String, KeyName As String, AttributeName As String)
    Sheets(Sh).Activate
    Dim Col_Key As Long
    Dim Col_Attr As Long
    Col_Key = z_GetColumnIndex(KeyName, 1, Sh)
    Col_Attr = z_GetColumnIndex(AttributeName, 1, Sh)
    Dim ColSize As Long
    Dim RowSize As Long
    ColSize = z_ColSize(1, Sh)
    RowSize = z_LastWrittenRow(Sh, ColSize)
    Dim Pild As String
    Dim Cnt As Long

    'loop through all rows
    For Row = 1 To RowSize
        If (Row Mod 1000) = 0 Then
            Debug.Print Row
        End If
        'find a new Pi

```

```

'Cells(Row, Col_Key).Select
Cnt = 1
If left(Cells(Row, Col_Key).Value, 2) = "PI" Then
    Cells(Row, Col_Attr).Value = Cnt
    Cnt = Cnt + 1
    Row = Row + 1
'Cells(Row, Col_Key).Select
'go through all rows of a Pi and count up
Do While Not left(Cells(Row, Col_Key).Value, 2) = "PI"
    Cells(Row, Col_Attr).Value = Cnt
    Cnt = Cnt + 1
    Row = Row + 1
'Cells(Row, Col_Key).Select
'termination criteria
If Row = RowSize + 1 Then
    Exit Function
End If
Loop
'Iterate down to remain at the same row after the next statement iteration
Row = Row - 1
End If
Next Row
End Function

```

Function z\_CopyWBSAttributeEntriesIntoResources(Sh As String, KeyName As String, AttributeName As String)

```

    Sheets(Sh).Activate
    Dim Col_Key As Long
    Dim Col_Attr As Long
    Col_Key = z_GetColumnIndex(KeyName, 1, Sh)
    Col_Attr = z_GetColumnIndex(AttributeName, 1, Sh)
    Dim ColSize As Long
    Dim RowSize As Long
    ColSize = z_ColSize(1, Sh)
    RowSize = z_LastWrittenRow(Sh, ColSize)

```

```

'loop through all rows
Dim Row As Long
Row = 1
Do While Row < RowSize + 1
    If (Row Mod 1000) = 0 Then
        Debug.Print Row
    End If
    'find a new Pi
    'Cells(Row, Col_Key).Select
    If left(Cells(Row, Col_Key).Value, 2) = "PI" Then
        'store the value of the attribute column on the Pi level
        Attr = Cells(Row, Col_Attr).Value
        Key = Cells(Row, Col_Key).Value
        Row = Row + 1
        'Cells(Row, Col_Key).Select
        'go through all rows of a Pi and write that stored value
        Do While Key = Cells(Row, Col_Key) And left(Cells(Row, Col_Attr).Value, 2) = Empty

```

```

        'write the value to the attr column of the ressources planed on the Pi level
        Cells(Row, Col_Attr).Value = Attr
        Row = Row + 1
        'Cells(Row, Col_Key).Select
        'termination criteria
        If Row = RowSize + 1 Then
            Exit Function
        End If
    Loop
    'find a new WS
    'Cells(Row, Col_Key).Select
    Row = Row - 1
    ElseIf left(Cells(Row, Col_Key).Value, 2) = "WS" Then
        'store the value of the attribute column on the WS level
        Attr = Cells(Row, Col_Attr).Value
        Key = Cells(Row, Col_Key).Value
        Row = Row + 1
        'Cells(Row, Col_Key).Select
        'go through all rows of a WS and write that stored value
        Do While Key = Cells(Row, Col_Key) And left(Cells(Row, Col_Attr).Value, 2) = Empty
            'write the value to the attr column of the ressources planed on the WS level
            Cells(Row, Col_Attr).Value = Attr
            Row = Row + 1
            'Cells(Row, Col_Key).Select
            'termination criteria
            If Row = RowSize + 1 Then
                Exit Function
            End If
        Loop
        'find a new TK
        'Cells(Row, Col_Key).Select
        Row = Row - 1
        ElseIf left(Cells(Row, Col_Key).Value, 2) = "TK" Then
            'store the value of the attribute column on the TK level
            Attr = Cells(Row, Col_Attr).Value
            Key = Cells(Row, Col_Key).Value
            Row = Row + 1
            'Cells(Row, Col_Key).Select
            'go through all rows of a TK and write that stored value
            Do While Key = Cells(Row, Col_Key) And left(Cells(Row, Col_Attr).Value, 2) = Empty
                'write the value to the attr column of the ressources planed on the TK level
                Cells(Row, Col_Attr).Value = Attr
                Row = Row + 1
                'Cells(Row, Col_Key).Select
                'termination criteria
                If Row = RowSize + 1 Then
                    Exit Function
                End If
            Loop
            'Cells(Row, Col_Key).Select
            Row = Row - 1
        Else
            'rows with Ressources or column names

```



```

    End If
    Row = Row + 1
Loop

End Function
Function z_MoveSheet(Sh As String, Wb_from As Workbook, Wb_to As Workbook, Where As String)

End Function

Function z_ColNamesExist(ColNames() As Variant, Sh As String)

End Function
Function z_ShMapColumns2(ByRef Arr_ColNames As Variant, ByRef Arr_ColNames_new As Variant, _
    Sh As String, Sh_new As String, Sh_log As String, _
    Sh_KeyCol As String, Sh_KeyCol_new As String, _
    Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Application.ScreenUpdating = False
Dim Start As Date: Dim Duration As Date
Start = Now() 'to measure the duration of the code
Dim ilog As Integer
ilog = 2

'Determine the column indizes in Sh and Sh_new,
Dim ColName_i As String
ReDim ColIndices(0 To UBound(Arr_ColNames), 0 To 2) As Variant

For i = LBound(Arr_ColNames) To UBound(Arr_ColNames) Step 1
    ColName_i = CStr(Arr_ColNames(i))
    ColIndices(i, 0) = ColName_i
    ColIndices(i, 1) = z_GetColumnIndex(ColName_i, 1, Sh)
    ColIndices(i, 2) = z_GetColumnIndex(ColName_i, 1, Sh_new)
    Debug.Print ColIndices(i, 0)
    Debug.Print ColIndices(i, 1)
    Debug.Print ColIndices(i, 2)
    If ColIndices(i, 1) = 0 Then
        'write errors into the logfile
        Sheets(Sh_log).Cells(ilog, 2) = "ColNames"
        Sheets(Sh_log).Cells(ilog, 3) = i
        Sheets(Sh_log).Cells(ilog, 4) = "not found in Sh"
        ilog = ilog + 1
        Stop 'make sure you find all!!!!!!
    ElseIf ColIndices(i, 2) = 0 Then
        'write errors into the logfile
        Sheets(Sh_log).Cells(ilog, 2) = "ColNames"
        Sheets(Sh_log).Cells(ilog, 3) = i
        Sheets(Sh_log).Cells(ilog, 4) = "not found in Sh_new"
        ilog = ilog + 1
        Stop 'make sure you find all!!!!!!
    End If
Next i

'Determine the row size in Sh_new

```

```

Sheets(Sh_new).Activate
RowSize_new = z_RowSize(1, Sh_new)

'Check whether column F is the "PIdentifier" in "Sh"
If z_PIdentifierCheck(Sh, 1, 6) = False Then
    Stop
End If
'Check whether column A is the "PIdentifier" and select Column A "PIdentifier" in "Sh_new"
If z_PIdentifierCheck(Sh_new, 1, 1) = False Then
    Stop
Else
    Range(Cells(2, 1), Cells(RowSize_new, 1)).Select
End If

'Iterate through the rows with "rcheck" = PIdentifier
Dim Row As Long
Row = 1
For Each rcheck In Selection.Cells
    If (Row Mod 1000) = 0 Then
        Debug.Print Row
    End If
    'if rcheck is found in Sh then perform the mapping
    If Not Sheets(Sh).Columns("F:F").Find(What:=rcheck, LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns
        For j = LBound(ColIndices) To UBound(ColIndices) Step 1
            'read the indices
            ColIndex_j = ColIndices(j, 1)
            ColIndex_new_j = ColIndices(j, 2)
            'map
            rcheck.Offset(0, (ColIndex_new_j) - 1).Value = _
                Sheets(Sh).Columns("F:F").Find(What:=rcheck, LookAt:=xlWhole).Offset(0, (ColIndex_j) -
6)
            Next j
        Else
            'write errors into the logfile
            Sheets(Sh_log).Cells(iLog, 2) = rcheck 'writes out PIdentifier
            Sheets(Sh_log).Cells(iLog, 4) = "not found"
        End If
        Row = Row + 1
    Next
    Sheets(Sh_new).Rows.RowHeight = 15
    'Write the durations into the logfile
    Duration = Now() - Start
    Sheets(Sh_log).Cells(iLog + 2, 1) = "Duration" & CStr(Duration): iLog = iLog + 1

Application.ScreenUpdating = True
Exit Function
NameExpectedNotExistent:
    'write into the RD_MasterDataSet_1
    Sheets("RD_MasterDataSet_1").Cells(1, EnumColIndexExp_i) = ColNameExp_i
    Sheets("RD_MasterDataSet_1").Cells(1, EnumColIndexExp_i).Font.Color = RGB(255, 255, 0)
    'write into the logfile
    iLog = iLog + 1

```

```

    Sheets("Logfile_MapPIsEAC").Cells(ilog, 2) = ColNameExp_i
    Sheets("Logfile_MapPIsEAC").Cells(ilog, 3) = "not found": ilog = ilog + 1
    NotfoundFlag = 0
    Resume Next
End Function

```

```

Function z_PidentifierCheck(Sh As String, Sh_row As Long, Sh_col As Long) As Boolean
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
If Sheets(Sh).Cells(Sh_row, Sh_col) = "Pidentifier" Then
    z_PidentifierCheck = True
Else
    z_PidentifierCheck = False
End If
End Function

```

```

Function z_CreateArrayInput_AttributeNames(Sh_from As String, Sh_to As String, NofNamesPerRow
As Integer)
'Input: Sh_from Row 1 containing all Attribute Names
'Output: Sh_to column A to column T containing "AttributeName1", "AttributeName2" ....
On Error Resume Next
WorksheetExists = (Sheets(Sh_to).Name <> "")
On Error GoTo 0
If WorksheetExists = False Then
    Worksheets.Add(Before:=Worksheets(1)).Name = Sh_to
End If

```

```

'Copy the Attribute Name from Sh_from to Sh_to
Sheets(Sh_from).Activate
Dim Row_from As Long
Dim Col_from As Long
Row_from = 1
Col_from_Size = z_ColSize(Row_from, Sh_from)
Dim Row_to As Long
Dim Col_to As Long
Row_to = 1
Col_to = 1
For Col_from = 1 To Col_from_Size
    Sheets(Sh_to).Cells(Row_to, Col_to) = Sheets(Sh_from).Cells(Row_from, Col_from)
    Col_to = Col_to + 1
    If Col_to Mod (NofNamesPerRow + 1) = 0 Then
        Col_to = 1
        Row_to = Row_to + 1
    End If
Next Col_from

```

```

'in Sh_to add "" and , and "_
Sheets(Sh_to).Activate
Row_to_Size = z_RowSize(1, Sh_to)
For Row_to = 1 To Row_to_Size
    For Col_to = 1 To NofNamesPerRow
        Sheets(Sh_to).Cells(Row_to, Col_to) = Chr(34) & CStr(Cells(Row_to, Col_to)) & Chr(34) & Chr(44)
    'Chr(34)=" Chr(44)=,
    Next Col_to

```

```

Next Row_to
For Row_to = 1 To Row_to_Size
    For Col_to = 1 To NofNamesPerRow
        If Col_to = NofNamesPerRow Then
            Sheets(Sh_to).Cells(Row_to, Col_to) = CStr(Cells(Row_to, Col_to)) & Chr(32) & Chr(95)
'Chr(32)=Space, Chr(95)=_
        End If
    Next Col_to
Next Row_to

End Function

Sub test12()
Call z_CreateArrayInput_AttributeNames("626d47cb89a01281d60a8f659070074",
"AttrNamesFromInput", 10)
End Sub

Function z_ShMapColumns(Sh_from As String, ColName_Key_from As String, ByRef ColNames_from
As Variant, _
    Sh_to As String, ColName_Key_to As String, ByRef ColNames_to As Variant, _
    Sh_log As String, Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Application.ScreenUpdating = False
'Start time measuring
Dim Start As Date: Dim Duration As Date
Start = Now()

'create a matrix with column names and indexes
MapMatrix = MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from,
ColNames_from, Sh_to, ColNames_to)

'Find the column index of the KeyName in "Sh_from"
Dim ColIndex_Key_from As Long
ColIndex_Key_from = z_GetColumnIndex(ColName_Key_from, 1, Sh_from)

'Find the column index of the KeyName in "Sh_to"
Dim ColIndex_Key_to As Long
ColIndex_Key_to = z_GetColumnIndex(ColName_Key_to, 1, Sh_to)

'Determine the row size in Sh_to
Sheets(Sh_to).Activate
RowSize_to = z_RowSize(ColIndex_Key_to, Sh_to)

'Select the range in the column Key_to
Sheets(Sh_to).Activate
Range(Cells(2, ColIndex_Key_to), Cells(RowSize_to, ColIndex_Key_to)).Select

'Iterate throught the rows with "rcheck" = Pidentifier
Dim ilog As Long
ilog = 2
For Each ValueInCol_Key_to In Selection.Cells
    'if ValueInCol_Key_to is found in Sh_from then perform the mapping

```

```

    If Not Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to,
LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns with the help of the MapMatrix
        For j = LBound(MapMatrix) To UBound(MapMatrix) Step 1
            'read the indices
            ColIndex_from_j = MapMatrix(j, 1)
            ColIndex_to_j = MapMatrix(j, 3)
            'map
            ValueInCol_Key_to.Offset(0, (ColIndex_to_j) - ColIndex_Key_to).Value = _
                Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to, _
                LookAt:=xlWhole).Offset(0, (ColIndex_from_j) - ColIndex_Key_from)
        Next j
    Else
        'write not found ValueInCol_Key_to in Sh_from into Sh_log
        Sheets(Sh_log).Cells(ilog, 2) = ValueInCol_Key_to
        Sheets(Sh_log).Cells(ilog, 4) = " not found, map them from another source file Sh_from"
        ilog = ilog + 1
    End If
Next

```

```

'In case the mapping has changed the row height
Sheets(Sh_to).Activate
Cells.Select
Selection.Rows.RowHeight = 15
Application.ScreenUpdating = True
'Write the durations into the logfile
Duration = Now() - Start
Sheets(Sh_log).Cells(ilog + 2, 1) = "Duration" & CStr(Duration)
End Function

```

```

Function z_ChkColExistence(Sh As String, ByRef ColNames As Variant, Sh_log As String) As Boolean
    'The column existence check is assumed to find all column names at the beginning
    z_ChkColExistence = True
    'Determine the column indices of the ColNames array in Sh
    Dim ColName_i As String
    ReDim Matrix_ColNameColIndex(0 To UBound(ColNames), 0 To 1) As Variant
    'iterate through the array
    For i = LBound(ColNames_from) To UBound(ColNames) Step 1
        ColName_i = CStr(ColNames(i))
        Matrix_ColNameColIndex(i, 0) = ColName_i
        Matrix_ColNameColIndex(i, 1) = z_GetColumnIndex(ColName_i, 1, Sh)
        Debug.Print Matrix_ColNameColIndex(i, 0) & " " & Matrix_ColNameColIndex(i, 1)
        If Matrix_ColNameColIndex(i, 1) = 0 Then
            'write errors into the logfile
            Sheets(Sh_log).Cells(ilog, 2) = "ColName: "
            Sheets(Sh_log).Cells(ilog, 3) = Matrix_ColNameColIndex(i, 0)
            Sheets(Sh_log).Cells(ilog, 4) = "not found in " & Sh
            ilog = ilog + 1
            'The column existence check has detected an unfound column name
            z_ChkColExistence = False
        End If
    Next i
End Function

```

```

Function MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from As String, ByRef
ColNames_from As Variant, _
    Sh_to As String, ByRef ColNames_to As Variant) As Variant
'Determine the column indizes in Sh and Sh_new,
Dim ColName_from_i As String
Dim ColName_to_i As String
ReDim Matrix_ColName1Index1_ColName2Index2(0 To UBound(ColNames_from), 0 To 3) As
Variant

For i = LBound(ColNames_from) To UBound(ColNames_from) Step 1
    ColName_from_i = CStr(ColNames_from(i))
    Matrix_ColName1Index1_ColName2Index2(i, 0) = ColName_from_i
    Matrix_ColName1Index1_ColName2Index2(i, 1) = z_GetColumnIndex(ColName_from_i, 1,
Sh_from)
    ColName_to_i = CStr(ColNames_to(i))
    Matrix_ColName1Index1_ColName2Index2(i, 2) = ColName_to_i
    Matrix_ColName1Index1_ColName2Index2(i, 3) = z_GetColumnIndex(ColName_to_i, 1, Sh_to)
    Debug.Print Matrix_ColName1Index1_ColName2Index2(i, 0) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 1) _
        & " " & Matrix_ColName1Index1_ColName2Index2(i, 2) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 3)
Next i
MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex =
Matrix_ColName1Index1_ColName2Index2
End Function

```

```

Sub testMapFkt()
'Clear the sheet Log1
Sheets("Log1").Activate
Cells.Select
Selection.ClearContents
Stop
'Clear the yellow cells in Sheet1
Sheets("Sheet1").Activate
Range(Cells(2, 3), Cells(22, 14)).ClearContents
Stop
'Define the Attributes from the source sheet Sh_from=Sheets2
Dim ColNames_from As Variant
ColNames_from = Array("Attr1", "Attr3", "Attr2", "Attr7", _
    "Attr9", "Attr5", "Attr8", "Attr4", "Attr6")
'Define the attributes from the target sheet Sh_to=Sheets1
'Even though the attribute names in Sh_from and Sh_to may be called differently they must
'refer to the same attribute and the have to be in the same order in both arrays!!!!
Dim ColNames_to As Variant
ColNames_to = Array("Attribute1", "Attribute3", "Attribute2", "Attribute7", _
    "Attribute9", "Attribute5", "Attribute8", "Attribute4", "Attribute6")
Call z_ShMapColumns("Sheet2", "Key1", ColNames_from, "Sheet1", "Key", ColNames_to, "Log1")
Stop
End Sub

```

## File: PERSONAL(2)

Public Function z\_OpenAndActivateWb(wbname As String, wbpas As String, ByRef Wb As Workbook)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Objective: Open the Workbook RD.xlsb if not already open and activate it.

'look if wbName is existent in workbooks list

Dim i As Long

For i = Workbooks.Count To 1 Step -1

    If Workbooks(i).Name = wbname Then Exit For

Next

'if wbName is existent in workbooks list, then i<>0-> activate workbook

'if wbName is not existent then i=0-> open workbook, activate workbook

If i <> 0 Then

    Set Wb = VBA.Interaction.GetObject(wbpas & wbname)

    Wb.Activate

Else

    Set Wb = Workbooks.Open(wbpas & wbname)

    Wb.Activate

End If

End Function

Function z\_WorkbookNewOrOpenOrActivate(wbname As String, wbpas As String, ByRef Wb As Workbook)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 19.10.2011

'Objective: Open the Workbook RD.xlsb if not already open and activate it.

'look if wbName is existent in workbooks list

Dim i As Long

For i = Workbooks.Count To 1 Step -1

    If Workbooks(i).Name = wbname Then Exit For

Next

'if wbName is existent in workbooks list, then i<>0-> activate workbook

'if wbName is not existent then i=0-> open workbook, activate workbook

If i <> 0 Then

    Set Wb = VBA.Interaction.GetObject(wbpas & wbname)

    Wb.Activate

Else

    On Error GoTo NewWB

    Set Wb = Workbooks.Open(wbpas & wbname)

    Wb.Activate

    On Error GoTo 0

End If

Exit Function

NewWB:

    Set Wb = Workbooks.Add

    Dim FileFormatValue As Integer

    If wbname <> Empty Then

        Select Case LCase(Right(wbname, Len(wbname) - InStrRev(wbname, ".", 1)))

```

        Case "xls": FileFormatValue = 56
        Case "xlsx": FileFormatValue = 51
        Case "xlsm": FileFormatValue = 52
        Case "xlsb": FileFormatValue = 50
        Case Else: FileFormatValue = 0
    End Select
End If
Wb.SaveAs Filename:=wbpath & wbname, FileFormat:=FileFormatValue
End Function

```

```

Function z_ShNewFlatValueCopy(Sh As String, Sh_new As String, Optional Where As String, Optional
ByRef Sh_Ref As Worksheet)

```

'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Name of the Sh to copy, Name of the new Sh\_new, where to place the new Sh\_new,  
' before or after some Sh\_Ref, at the begin or the end

```

    Call z_ShAdd(Where, Sh_Ref)
    On Error Resume Next
    ActiveSheet.Name = Sh_new
    If Err.Number <> 0 Then
        Application.DisplayAlerts = False
        ActiveSheet.Delete
        Application.DisplayAlerts = True
        Sheets(Sh_new).Cells.ClearContents
    End If
    On Error GoTo 0
    Sheets(Sh).Cells.Copy
    Sheets(Sh_new).Select
    Sheets(Sh_new).Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
    Sheets(Sh_new).Columns("A:GA").ColumnWidth = 20
End Function

```

```

Function z_ShDelete(Sh As String, Optional ByRef Wb As Workbook)

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

```

    Wb.Activate
    Application.DisplayAlerts = False
    On Error Resume Next
    Sheets(Sh).Delete
    On Error GoTo 0
    Application.DisplayAlerts = True
End Function

```

```

Function z_ExcelSessionWindowNormal(Optional ByRef Wb As Workbook)

```

'Fenster der Excel session

```

    If Wb Is Nothing Then
        Application.WindowState = xlNormal
    Else
        Wb.Application.WindowState = xlNormal
        Wb.Activate
    End If
End Function

```



```
Function z_ExcelSessionWindowMoveAndResize(Optional ByRef Wb As Workbook, _
    Optional top As Variant, Optional left As Variant, _
    Optional width As Variant, Optional height As Variant)
```

```
    If Wb Is Nothing Then
        If top <> Empty Then
            Application.top = top
        End If
        If left <> Empty Then
            Application.left = left
        End If
        If width <> Empty Then
            Application.width = width
        End If
        If height <> Empty Then
            Application.height = height
        End If
```

```
    Else
        If top <> Empty Then
            Wb.Application.top = CInt(top)
        End If
        If left <> Empty Then
            Wb.Application.left = CInt(left)
        End If
        If width <> Empty Then
            Wb.Application.width = CInt(width)
        End If
        If height <> Empty Then
            Wb.Application.height = CInt(height)
        End If
    End If
```

```
End Function
```

```
Function z_ShNew(Sh As String, Optional Where As String, Optional ByRef Sh_Ref As Worksheet,
    Optional ByRef Wb As Workbook)
```

```
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

```
'Date: 26.9.2011
```

```
'Input: Name of the new Sh, where to place the new Sh, before or after some Sh_Ref, at the begin or
the end
```

```
On Error Resume Next
```

```
WorksheetExists = (Sheets(Sh).Name <> "")
```

```
On Error GoTo 0
```

```
If WorksheetExists = False Then
```

```
    Call z_ShAdd(Where, Sh_Ref)
```

```
    ActiveSheet.Name = Sh 'Worksheets.Add(Before:=Worksheets(1)).Name = Sh
```

```
End If
```

```
'Clear contents
```

```
Sheets(Sh).Activate
```

```
ActiveSheet.Cells.Select
```

```
Selection.ClearContents
```

```
'Format
```

```
Sheets(Sh).Columns.ColumnWidth = 20
```

```
Sheets(Sh).Rows.RowHeight = 15
```

End Function

Function z\_ShAdd(Optional Where As String, Optional ByRef Sh\_Ref As Worksheet)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: where to place the added Sh, before or after some Sh\_Ref, at the begin or the end

If Not Sh\_Ref Is Nothing Then

    If Where = ("Before:=") Then

        Sheets.Add Sh\_Ref

    Elseif Where = ("After:=") Then

        Sheets.Add , Sh\_Ref

    Else

        Sheets.Add Before:=Sheets(1)

    End If

Else

    If Where = ("End") Then

        Sheets.Add After:=Sheets(Sheets.Count)

    Elseif Where = ("Begin") Then

        Sheets.Add Before:=Sheets(1)

    Else

        Sheets.Add Before:=Sheets(1)

    End If

End If

End Function

Public Function z\_GetColumnIndex2(ByRef SearchString As String, SearchRow As Integer, \_  
    Optional Sh As String, Optional ByRef Wb As Workbook) As Long

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Output datatype change from Variant

Dim CellIndexStr As String 'In R1C1 Format

Dim CellIndexArr() As String 'Splited R1C1 Format

Dim ColIndex As Integer

'Activate the right Wb and Sh

On Error GoTo OptionalArgument:

Wb.Activate

On Error GoTo 0

Sheets(Sh).Activate

'find column name

Dim Cl As Range

On Error GoTo NameExpectedNotExist:

Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole).Select

On Error GoTo 0

'find column index

CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)

CellIndexArr = Split(CellIndexStr, "C")

ColIndex = CInt(CellIndexArr(1))

'Output

z\_GetColumnIndex2 = ColIndex

Exit Function

OptionalArgument:

    Resume Next

NameExpectedNotExist:

    ColIndex = 0

```

    Stop 'only in test mode
    z_GetColumnIndex2 = ColIndex
End Function

```

```

Public Function z_GetColumnIndex3(ByRef SearchString As String, SearchRow As Integer, _
    Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Output datatype change from Variant
Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer
'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
'Sheets(Sh).Activate
'find column name
Dim Cl As Range
Set Cl = Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole)
If Cl Is Nothing Then GoTo NameExpectedNotExistent
'find column index
'CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexStr = Cl.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
'Output
z_GetColumnIndex3 = ColIndex
Exit Function
OptionalArgument:
Resume Next
NameExpectedNotExistent:
ColIndex = 0
Stop 'only in test mode
z_GetColumnIndex3 = ColIndex
End Function

```

```

Public Function z_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, _
    Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Output datatype change from Variant
Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer
'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
'Sheets(Sh).Activate
Dim Sht As Worksheet
Set Sht = Sheets(Sh)
'find column name
Dim Cl As Range
'Set Cl = Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole)

```

```

Set Cl = Sht.Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole)
If Cl Is Nothing Then GoTo NameExpectedNotExistent
'find column index
'CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexStr = Cl.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = Clnt(CellIndexArr(1))
'Output
z_GetColumnIndex = ColIndex
Exit Function
OptionalArgument:
Resume Next
NameExpectedNotExistent:
ColIndex = 0
Stop 'only in test mode
z_GetColumnIndex = ColIndex
End Function

Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet
'Sheets(Sh).Activate
'Determine the row size
z_RowSize = If(IsEmpty(Sheets(Sh).Cells(1048576, SearchCol)), Sheets(Sh).Cells(1048576, SearchCol).End(xlUp).Row, 1048576)
End Function

Function z_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Row, Output: column with the last entry in that row
'SearchCol datatype changed from integer
'Activate the Sheet
'Sheets(Sh).Activate
'Determine the col size
z_ColSize = If(IsEmpty(Sheets(Sh).Cells(SearchRow, 16384)), Sheets(Sh).Cells(SearchRow, 16384).End(xlToLeft).Column, 16384)
End Function

Function z_CopyRange(PasteAllOrValuesOrFormats As String, Sh_from As String, Range_From As Range, _
                    Sh_to As String, Cell_to As Range)
'only works out if you invoke this function after this line!:Sheets(Sh_from).Activate
'otherwise VBA cannot set the Range_From
'set Cell_to as follows: Cell_to=Sheets(Sh_to).Range("A1")
Sheets(Sh_from).Activate
Range_From.Select
Selection.Copy

```

```

Sheets(Sh_to).Activate
Cell_to.Select
Dim first As Long
Dim last As Long
If PasteAllOrValuesOrFormats = "All" Then
    first = Range_From.Columns.End(xlToLeft).Column
    last = Range_From.Columns.End(xlToRight).Column
    Call z_Copy_ColWidth(Sh_from, Sh_to, first, last)
    Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
ElseIf PasteAllOrValuesOrFormats = "Values" Then
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
ElseIf PasteAllOrValuesOrFormats = "Formats" Then
    Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False
End If
End Function

Function z_Copy_ColWidth(Sh_from As String, Sh_to As String, Col_From, Col_to)
    For Col = Col_From To Col_to
        Sheets(Sh_to).Columns(Col).ColumnWidth = Sheets(Sh_from).Columns(Col).ColumnWidth
    Next Col
End Function

Function z_ChangeColWidth(ColWidth As Double, Sh As String, Optional Col_From As Long, Optional
Col_to As Long)
    If Col_From = 0 Then
        Col_From = 1
    End If
    If Col_to = 0 Then
        Col_to = 16384
    End If
    For Col = Col_From To Col_to
        If Sheets(Sh).Columns(Col).ColumnWidth <> ColWidth Then
            Sheets(Sh).Columns(Col).ColumnWidth = ColWidth
        End If
    Next Col
End Function

Function z_ChangeRowHeight(RowHeight As Double, Sh As String, Optional Row_from As Long,
Optional Row_to As Long)
    If Row_from = 0 Then
        Row_from = 1
    End If
    If Row_to = 0 Then
        Row_to = 1048576
    End If
    For Row = Row_from To Row_to
        If Sheets(Sh).Rows(Row).RowHeight <> RowHeight Then
            Sheets(Sh).Rows(Row).RowHeight = RowHeight
        End If
    Next Row

```

End Function

```
Sub z_CopySheet(Sh_from As String, Sh_to As String)
    Sheets(Sh_from).Select
    Cells.Select
    Application.CutCopyMode = False
    Selection.Copy
    Sheets(Sh_to).Select
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _
        False, Transpose:=False
    Range("A1").Select
    Sheets(Sh_from).Select
    Range("A1").Select
End Sub
```

```
Function z_CopyRow(Sh_from As String, Row_from As Long, Sh_to As String, Row_to As Long)
    Sheets(Sh_from).Activate
    Sheets(Sh_from).Rows(Row_from).Select
    Selection.Copy
    Sheets(Sh_to).Activate
    Sheets(Sh_to).Rows(Row_to).Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
End Function
```

```
Function z_InsertRow(Sh_from As String, Row_from As Long, Sh_to As String, Row_to As Long)
    Sheets(Sh_from).Activate
    Sheets(Sh_from).Rows(Row_from).Select
    Selection.Copy
    Sheets(Sh_to).Activate
    Sheets(Sh_to).Rows(Row_to).Select
    Selection.Insert Shift:=xlDown
End Function
```

```
Function z_CopyColumn(Sh_from As String, Col_From As Long, Sh_to As String, Col_to As Long)
    Sheets(Sh_from).Activate
    Sheets(Sh_from).Columns(Col_From).Select
    Selection.Copy
    Sheets(Sh_to).Activate
    Sheets(Sh_to).Columns(Col_to).Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
End Function
```

```
Function z_InsertColumn(Sh_from As String, Col_From As Long, Sh_to As String, Col_to As Long)
    Sheets(Sh_from).Activate
    Sheets(Sh_from).Columns(Col_From).Select
    Selection.Copy
    Sheets(Sh_to).Activate
    Sheets(Sh_to).Columns(Col_to).Select
    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
End Function
```

```

Function z_InsertEmptyCols(Sh As String, nofCols As Integer, Col_to As Long)
    Sheets(Sh).Columns(Col_to).Select
    For iter = 1 To nofCols
        Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
    Next iter
End Function

```

```

Function z_InsertEmptyRows(Sh As String, NofRows As Integer, Row_to As Long)
    Sheets(Sh).Rows(Row_to).Select
    For iter = 1 To NofRows
        Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
    Next iter
End Function

```

```

Function z_ClearRowContents(Sh As String, Row As Long)
    Sheets(Sh).Rows(Row).Select
    Selection.ClearContents
End Function

```

```

Function z_ClearColContents(Sh As String, Col As Long)
    Sheets(Sh).Columns(Col).Select
    Selection.ClearContents
End Function

```

```

Function z_DeleteRowsOfNotEmptyCells(Sh As String, Col As Long, FirstRow As Long)
    RowSize = z_RowSize(1, Sh)
    For Row = FirstRow To RowSize
        Sheets(Sh).Cells(Row, Col).EntireRow.Select
        If Sheets(Sh).Cells(Row, Col) <> Empty Then
            Sheets(Sh).Cells(Row, Col).EntireRow.Delete
            Row = Row - 1
        End If
    Next Row
End Function

```

```

Function z_DeleteRows(Sh As String, Row_from As Long, Row_to As Long)
    Cells(Row_from, 1).Select
    For Row = Row_from To Row_to
        Sheets(Sh).Rows(Row_from).EntireRow.Delete
    Next Row
End Function

```

```

Private Sub test23()
    Call z_DeleteRows("RD_MasterDataSet_1", 5, 7)
End Sub

```

```

Function z_Sort(Sh As String, Col1 As Long, Col2 As Long)
    RowSize = z_RowSize(1, Sh)
    ColSize = z_ColSize(1, Sh)
    Sheets(Sh).Sort.SortFields.Clear
    Sheets(Sh).Sort.SortFields.Add Key:=Range(Cells(2, Col1), Cells(RowSize, Col1)), _
        SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal

```

```

Sheets(Sh).Sort.SortFields.Add Key:=Range(Cells(2, Col2), Cells(RowSize, Col2)), _
    SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
With ActiveWorkbook.Worksheets(Sh).Sort
    .SetRange Range(Cells(1, 1), Cells(RowSize, ColSize))
    .Header = xlYes
    .MatchCase = False
    .Orientation = xlTopToBottom
    .SortMethod = xlPinYin
    .Apply
End With
End Function

Function z_WorkbookSave(Wb As Workbook)
    Wb.Save
End Function

Function z_MoveSheetFromWb1ToWb2(Sh_from As String, Wb_from As Workbook, Wb_to As
Workbook, Where As String)
    Windows(Wb_from.Name).Activate
    Sheets(Sh_from).Activate
    Sheets(Sh_from).Move Before:=Wb_to.Sheets(1)
End Function

Function z_CopySheetFromWb1ToWb2(Sh_from As String, Wb_from As Workbook, Wb_to As
Workbook, Where As String)
    Windows(Wb_from.Name).Activate
    Sheets(Sh_from).Activate
    Sheets(Sh_from).Copy Before:=Wb_to.Sheets(1)
End Function

Function z_MoveWbSheetsIntoAnotherWb(Wb_ref As Workbook)
    Dim Wb As Workbook
    For Each Wb In Wb_ref.Application.Workbooks ' Workbooks
        'move all Windows to Wb_ref
        If Wb.Name <> Wb_ref.Name Then
            If Wb.Name <> "PERSONAL.XLSB" Then
                If Wb.Name <> "API_VBA_GenerateSmartchoiceReports.xlsb" Then
                    Windows(Wb.Name).Activate
                    Sheets(left(Wb.Name, 31)).Select
                    Sheets(left(Wb.Name, 31)).Move After:=Wb_ref.Sheets(Sheets.Count)
                End If
            End If
        End If
    Next
    'For Each Wb In Wb_ref.Application.Workbooks
    '    If Wb.Name = "PERSONAL.XLSB" Then
    '        Windows(Wb.Name).Visible = False
    '    End If
    'Next
End Function

Private Sub test()
    Dim ColStart As Long

```



```
ColStart = z_ColSize(1, "RD_MasterDataSet_5") + 1
Call z_AddColNames(Array("Test1", "Test2"), "RD_MasterDataSet_5", 1)
End Sub
```

```
Function z_AddColNames(ByRef ColNames As Variant, Optional Sh As String, Optional Row As Integer
= 1, Optional Col_Start As Long, Optional ByRef Wb As Workbook)
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
If Col_Start = 0 Then
    Size_ColNames = UBound(ColNames)
    For j = Col_Start To Size_ColNames
        Cells(Row, j + 1) = ColNames(j)
    Next j
Else
    Size_ColNames = UBound(ColNames) + Col_Start
    For j = Col_Start To Size_ColNames
        Cells(Row, j) = ColNames(j - Col_Start)
    Next j
End If
Cells.Select
'Selection.EntireColumn.AutoFit
Selection.ColumnWidth = 20
End Function
```

```
Function z_ChgFmt_CostCols(Sh As String, FromCol As Long, ToCol As Long)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 27.9.2011
Sheets(Sh).Select
RowSize = z_RowSize(1, Sh)
Range(Cells(2, FromCol), Cells(RowSize, ToCol)).Select
'Rows("1:1").Find(What:="EtcTrialsFullCosts2010", LookAt:=xlWhole).Select
'Range(ActiveCell.End(xlToRight).Offset(1, -2), ActiveCell.End(xlDown)).Select 'offset?
'Range(ActiveCell.End(xlToRight), ActiveCell.End(xlDown)).Select
Selection.NumberFormat = "#,##0"
Selection.Replace What:=".", Replacement:=".", LookAt:=xlPart, _
    SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
    ReplaceFormat:=False
Selection.ColumnWidth = 25
End Function
```

```
Function z_CopyInsertRange2(Range_From As Range, Range_To As Range, Sh_from As String, Sh_to
As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Copy a range (e.g all entries of a column) from Sh_from to a defined cell in Sh_to
Sheets(Sh_from).Select
Range_From.Copy
Sheets(Sh_to).Select
Range_To.Insert
End Function
```

```
Function z_CopyInsertRange(RowLU_From As Long, ColLU_From As Long, RowRD_From As Long,
ColRD_From As Long, Sh_from As String, _
    RowLU_To As Long, ColLU_To As Long, Sh_to As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

```

'Copy a range (e.g all entries of a column) from Sh_from to a defined cell in Sh_to
Sheets(Sh_from).Select
Range(Cells(RowLU_From, ColLU_From), Cells(RowRD_From, ColRD_From)).Select
Selection.Copy
Sheets(Sh_to).Select
Cells(RowLU_To, ColLU_To).Insert 'moves the other columns to the right
End Function

```

```

Function z_CopyPasteRange2(Range_From As Range, Range_To As Range, Sh_from As String, Sh_to
As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'copy the Range_from in Sh_from and paste it to Range_to in Sh_to
Sheets(Sh_from).Range_From.Copy Destination:=Sheets(Sh_to).Range_To
End Function

```

```

Function z_CopyPasteRange(RowLU_From As Long, ColLU_From As Long, RowRD_From As Long,
ColRD_From As Long, Sh_from As String, _
RowLU_To As Long, ColLU_To As Long, Sh_to As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'copy the Range_from in Sh_from and paste it to Range_to in Sh_to
Sheets(Sh_from).Select
Range(Cells(RowLU_From, ColLU_From), Cells(RowRD_From, ColRD_From)).Copy _
Destination:=Sheets(Sh_to).Cells(RowLU_To, ColLU_To)
End Function

```

```

Function z_LastWrittenRow(Optional Sh As String, Optional StartAtCol As Long, _
Optional StopAtCol As Long, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 11.10.2011
'Input: All Columns, Output: last written row
'FirstEmptyCol = z_FirstEmptyCol()
Dim RowSize_max As Long: RowSize_max = 0
Dim RowSize_col As Long: RowSize_col = 0
Dim AllCols As Long: AllCols = 16384
Dim Col As Long
'Optional input
If StartAtCol = 0 Then
StopAtCol = 1
End If
If StopAtCol = 0 Then
StopAtCol = AllCols
End If
For Col = StartAtCol To StopAtCol
RowSize_col = z_RowSize(Col, Sh)
If RowSize_col > RowSize_max Then
RowSize_max = RowSize_col
End If
Next Col
z_LastWrittenRow = RowSize_max
End Function

```

```

Function z_LastWrittenRowAndCol(Optional Sh As String, Optional StopAtRow As Long, _
Optional StopAtCol As Long, Optional ByRef Wb As Workbook) As Variant

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 11.10.2011

'Input:(sh,StopAtRow) or (sh,StopAtRow,0)

    'find ColSize\_max from 1 to StopAtRow

    'find RowSize\_max from 1 to ColSize\_max

'Input:(sh, ,StopAtCol) or (sh,0,StopAtRow)

    'find RowSize\_max from 1 to StopAtCol

    'find ColSize\_max from 1 to RowSize\_max

'Input:(sh) or (sh,0,0)

    'find RowSize\_max from 1 to 16384

    'find ColSize\_max from 1 to RowSize\_max

Dim ColSize\_max As Long: ColSize\_max = 0

Dim ColSize\_row As Long: ColSize\_row = 0

Dim Row As Long

'Optional input

Dim StopAtRow\_tmp As Long: StopAtRow\_tmp = StopAtRow

If StopAtRow = 0 Then

    StopAtRow = z\_LastWrittenRow(Sh, 1, StopAtCol)

End If

For Row = 1 To StopAtRow

    ColSize\_row = z\_ColSize(Row, Sh)

    If ColSize\_row > ColSize\_max Then

        ColSize\_max = ColSize\_row

    End If

Next Row

If StopAtRow\_tmp <> 0 Then

    RowSize\_max = z\_LastWrittenRow(Sh, 1, ColSize\_max)

End If

Dim out(0 To 1) As Variant

out(0) = RowSize\_max 'The Last written row if StopAtCol is the last written column

out(1) = ColSize\_max 'The last written col if StopAtRow is the last written row

z\_LastWrittenRowAndCol = out

End Function

Function z\_CellToIndex(ByRef Cell\_in As Range) As Variant

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Dim CellIndexStr As String 'In R1C1 Format

Dim CellIndexArr() As String 'Splited R1C1 Format

Dim ColIndex As Integer

Dim RowIndex As Integer

Dim CellIndices(0 To 1) As Long

'find column index

CellIndexStr = Cell\_in.Address(ReferenceStyle:=xlR1C1)

CellIndexArr = Split(CellIndexStr, "C")

ColIndex = CInt(CellIndexArr(1))

CellIndexArr = Split(CellIndexArr(0), "R")

RowIndex = CInt(CellIndexArr(1))

CellIndices(0) = RowIndex

CellIndices(1) = ColIndex

'Output

z\_CellToIndex = CellIndices

End Function

\*\*\*\*\*Get Range Indices

Function z\_RangeToIndices(ByRef Rng As Range) As Variant

'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Dim RangeIndices(0 To 3) As Long

Dim CellsArray() As String

Dim sAddr As String

sAddr = Rng.Address(ReferenceStyle:=xlR1C1)

CellsArray = Split(sAddr, ":")

Dim CellIndicesUL() As Long

On Error GoTo RangelsColumnOrRow

CellIndicesUL = z\_sCellToIndex(CellsArray(0))

On Error GoTo 0

Dim CellIndicesLR() As Long

On Error GoTo RangelsCell

CellIndicesLR = z\_sCellToIndex(CellsArray(1))

On Error GoTo 0

RangeIndices(0) = CellIndicesUL(0)

RangeIndices(1) = CellIndicesUL(1)

RangeIndices(2) = CellIndicesLR(0)

RangeIndices(3) = CellIndicesLR(1)

z\_RangeToIndices = RangeIndices

Exit Function

RangelsCell:

CellIndicesLR = z\_sCellToIndex(CellsArray(0))

Resume Next

RangelsColumnOrRow:

Dim RorC As String

RorC = left(sAddr, 1)

OneOrMore = InStr(1, sAddr, ":", vbTextCompare)

'only one row or column

If OneOrMore = 0 Then

If RorC = "C" Then

RangeIndices(0) = 1

RangeIndices(1) = z\_sColumnToIndex(CellsArray(0))

RangeIndices(2) = 1048534

RangeIndices(3) = RangeIndices(0)

Elseif RorC = "R" Then

RangeIndices(0) = z\_sRowToIndex(CellsArray(0))

RangeIndices(1) = 1

RangeIndices(2) = RangeIndices(0)

RangeIndices(3) = 16383

Else

Stop

End If

'more than one row or column

Else

If RorC = "C" Then

RangeIndices(0) = 1

RangeIndices(1) = z\_sColumnToIndex(CellsArray(0))

RangeIndices(2) = 1048534

```

        RangeIndices(3) = z_sColumnToIndex(CellsArray(1))
Elseif RorC = "R" Then
    RangeIndices(0) = z_sRowToIndex(CellsArray(0))
    RangeIndices(1) = 1
    RangeIndices(2) = z_sRowToIndex(CellsArray(1))
    RangeIndices(3) = 16383
Else
    Stop
End If
End If
z_RangeToIndices = RangeIndices
End Function
Function z_sCellToIndex(ByRef CellIndexStr As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer
Dim RowIndex As Integer
Dim CellIndices(0 To 1) As Long
'find column index
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
CellIndexArr = Split(CellIndexArr(0), "R")
RowIndex = CInt(CellIndexArr(1))
CellIndices(0) = RowIndex
CellIndices(1) = ColIndex
'Output
z_sCellToIndex = CellIndices
End Function
Function z_sColumnToIndex(ByRef ColIndexStrLeft As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Dim ColArray() As String
ColArray = Split(ColIndexStrLeft, "C")
z_sColumnToIndex = ColArray(1)
End Function
Function z_sRowToIndex(ByRef RowIndexStrUp As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Dim RowArray() As String
RowArray = Split(RowIndexStrUp, "R")
z_sRowToIndex = RowArray(1)
End Function
*****Get Range Indices

Function z_IndicesToRange(RowUL As Long, ColUL As Long, RowDR As Long, ColDR As Long) As Range
Dim Rng As Range
Rng = Range(Cells(RowUL, ColUL), Cells(RowDR, ColDR))
End Function

Function z_TrimRow(Sh As String, Col_From As Long, Col_to As Long, Optional CheckRow As Long)
Sheets(Sh).Activate
Dim Rng As Range
If CheckRow = 0 Then
    Cells.Select
    Set Rng = Selection.Cells

```

```

Else
    Set Rng = Range(Cells(CheckRow, Col_From), Cells(CheckRow, Col_to))
End If

For Each Cell In Rng
    Cell.Value = Excel.WorksheetFunction.Trim(WorksheetFunction.Clean(Cell))
Next Cell
End Function

Function z_TrimCells(Sh As String, Optional ByRef Rng As Range)
'Sheets("Sheet1").Activate
'Call z_TrimCells("Sheet1", Range(Cells(3, 1), Cells(3, 5)))

Sheets(Sh).Activate
If Rng Is Nothing Then
    'select all
    Cells.Select
    Set Rng = Selection.Cells
Else
    'take the input range
End If

Dim MyRngIndices() As Long
MyRngIndices = z_RangeToIndices(Rng)

For Row = MyRngIndices(0) To MyRngIndices(2)
    For Col = MyRngIndices(1) To MyRngIndices(3)
        With Excel.WorksheetFunction
            Cells(Row, Col) = .Trim(.Clean(Cells(Row, Col)))
        End With
    Next Col
Next Row
End Function

Function z_SelectMultiple(sSh As String, SearchCol As Long, SearchText As String)
Dim Sh As Worksheet
Dim Rng As Range
Dim rngFind As Range
Dim firstAddress As String
Dim addSelection As String
Set Sh = Worksheets(sSh)
' Set our range to search
RowSize = z_RowSize(SearchCol, sSh)
Set Rng = Sh.Range(Cells(2, SearchCol), Cells(RowSize, SearchCol))
'
With Rng
    ' Find our required text
    Set rngFind = .Find(SearchText)
    ' If we find it then...
    If Not rngFind Is Nothing Then
        firstAddress = rngFind.Address ' Take a note of where we first found it
        addSelection = addSelection & rngFind.Address & ", " ' Add the cell's 'range to our selection
        ' Loop through the rest of our range and find any other instances.
    End If
End With

```

```

    Do
        Set rngFind = .FindNext(rngFind)
        addSelection = addSelection & rngFind.Address & ","
    Loop While Not rngFind Is Nothing And rngFind.Address <> firstAddress

    End If
End With
' Trim the last comma from our string
addSelection = Mid(addSelection, 1, Len(addSelection) - 1)
Dim SelectionSplit As Variant
SelectionSplit = Split(addSelection, ",")
Dim sSelection As String
sSelection = CStr(SelectionSplit(0)) & ":" & CStr(SelectionSplit(UBound(SelectionSplit) - 1))
Sh.Range(sSelection).Select ' Select our rows!
Set Rng = Nothing
Set Sh = Nothing
End Function

Function z_SelectMultiple2(sSh As String, SearchCol As Long, SearchText As String, ByRef RngOut As Range)
    Dim Sh As Worksheet
    Dim Rng As Range
    Dim rngFind As Range
    Dim firstAddress As String
    Dim addSelection As String
    Set Sh = Worksheets(sSh)
    ' Set our range to search
    RowSize = z_RowSize(SearchCol, sSh)
    Set Rng = Sh.Range(Cells(2, SearchCol), Cells(RowSize, SearchCol))
    ' Find our first required text with Find
    Set rngFind = Rng.Find(SearchText)
    ' If we find it then...
    If Not rngFind Is Nothing Then
        firstAddress = rngFind.Address ' Take a note of where we first found it
        addSelection = addSelection & rngFind.Address & "," ' Add the cell's 'range to our selection
        ' Loop through the rest of our range and find any other instances with FindNext.
        Do
            Set rngFind = Rng.FindNext(rngFind)
            addSelection = addSelection & rngFind.Address & ","
        Loop While Not rngFind Is Nothing And rngFind.Address <> firstAddress
    End If
    ' Trim the last comma from our string
    addSelection = Mid(addSelection, 1, Len(addSelection) - 1)
    Dim SelectionSplit As Variant
    SelectionSplit = Split(addSelection, ",")
    Dim sSelection As String
    'sSelection = CStr(CStr(SelectionSplit(UBound(SelectionSplit) - 1) & ":" & SelectionSplit(0)))
    sSelection = CStr(SelectionSplit(0)) & ":" & CStr(SelectionSplit(UBound(SelectionSplit) - 1))
    'Sh.Range(sSelection).Select ' Select our rows!
    Set RngOut = Sh.Range(sSelection)
    'RngOut.Select
    Set Rng = Nothing
    Set Sh = Nothing
End Function

```

End Function

Function z\_CopySh1ToSh2\_GivenColLastRow(Sh\_from As String, ColNameStart\_from As String, Sh\_to As String)

```
Dim ColStart_from As Long
ColStart_from = z_GetColumnIndex(ColNameStart_from, 1, Sh_from)
Dim ColSize_from As Long
ColSize_from = z_ColSize(1, Sh_from)
'RowSize_from = z_RowSize(1, Sh_from)
RowSize_from = z_LastWrittenRow(Sh_from, 1, ColSize_from)
Sheets(Sh_from).Activate
Range(Cells(1, ColStart_from), Cells(RowSize_from, ColSize_from)).Select
Selection.Copy
'RowSize_to = z_RowSize(1, Sh_to) + 1
RowSize_To = z_LastWrittenRow(Sh_to, 1, ColSize_from) + 1
Sheets(Sh_to).Activate
Sheets(Sh_to).Cells(RowSize_To, 1).Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
```

End Function

Function z\_ShMapColumns(Sh\_from As String, ColName\_Key\_from As String, ByRef ColNames\_from As Variant, \_

Sh\_to As String, ColName\_Key\_to As String, ByRef ColNames\_to As Variant, \_

Optional Sh\_log As String, Optional ByRef Wb As Workbook, \_

Optional KeyRow\_from As Integer, Optional KeyRow\_to As Integer)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Application.ScreenUpdating = False

'Start time measuring

Dim Start As Date: Dim Duration As Date

Start = Now()

'optional KeyRows

If KeyRow\_from = 0 Then

KeyRow\_from = 1

End If

If KeyRow\_to = 0 Then

KeyRow\_to = 1

End If

'create a matrix with column names and indexes

```
MapMatrix = MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from,
ColNames_from, Sh_to, ColNames_to, _
KeyRow_from, KeyRow_to)
```

'Find the column index of the KeyName in "Sh\_from"

Dim ColIndex\_Key\_from As Long

ColIndex\_Key\_from = z\_GetColumnIndex(ColName\_Key\_from, KeyRow\_from, Sh\_from)

'Find the column index of the KeyName in "Sh\_to"

Dim ColIndex\_Key\_to As Long

ColIndex\_Key\_to = z\_GetColumnIndex(ColName\_Key\_to, KeyRow\_to, Sh\_to)



```

'Determine the row size in Sh_to
Sheets(Sh_to).Activate
RowSize_To = z_RowSize(ColIndex_Key_to, Sh_to)

'Select the range in the column Key_to
Sheets(Sh_to).Activate
Range(Cells(2, ColIndex_Key_to), Cells(RowSize_To, ColIndex_Key_to)).Select

'Iterate through the rows with "rcheck" = PIdentifier
Dim ilog As Long
ilog = 2
For Each ValueInCol_Key_to In Selection.Cells
    'if ValueInCol_Key_to is found in Sh_from then perform the mapping
    If Not Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to,
LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns with the help of the MapMatrix
        For j = LBound(MapMatrix) To UBound(MapMatrix) Step 1
            'read the indices
            ColIndex_from_j = MapMatrix(j, 1)
            ColIndex_to_j = MapMatrix(j, 3)
            'map
            ValueInCol_Key_to.Offset(0, (ColIndex_to_j) - ColIndex_Key_to).Value = _
                Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to, _
                LookAt:=xlWhole).Offset(0, (ColIndex_from_j) - ColIndex_Key_from)
        Next j
    Else
        'write not found ValueInCol_Key_to in Sh_from into Sh_log
        'Sh_log is optional, if not existent as input resume next
        On Error Resume Next
        Sheets(Sh_log).Cells(ilog, 2) = ValueInCol_Key_to
        Sheets(Sh_log).Cells(ilog, 4) = " not found, map them from another source file Sh_from"
        ilog = ilog + 1
        On Error GoTo 0
    End If
Next

'In case the mapping has changed the row height
Sheets(Sh_to).Activate
Cells.Select
Selection.Rows.RowHeight = 15
Application.ScreenUpdating = True
'Write the durations into the logfile
'Sh_log is optional, if not existent as input resume next
Duration = Now() - Start
On Error Resume Next
Sheets(Sh_log).Cells(ilog + 2, 1) = "Duration" & CStr(Duration)
On Error GoTo 0
End Function

Function z_ChkColExistence(Sh As String, ByRef ColNames As Variant, Sh_log As String) As Boolean
    'The column existence check is assumed to find all column names at the beginning
    z_ChkColExistence = True
    'Determine the column indices of the ColNames array in Sh

```

```

Dim ColName_i As String
ReDim Matrix_ColNameColIndex(0 To UBound(ColNames), 0 To 1) As Variant
'iterate through the array
For i = LBound(ColNames_from) To UBound(ColNames) Step 1
    ColName_i = CStr(ColNames(i))
    Matrix_ColNameColIndex(i, 0) = ColName_i
    Matrix_ColNameColIndex(i, 1) = z_GetColumnIndex(ColName_i, 1, Sh)
    Debug.Print Matrix_ColNameColIndex(i, 0) & " " & Matrix_ColNameColIndex(i, 1)
    If Matrix_ColNameColIndex(i, 1) = 0 Then
        'write errors into the logfile
        Sheets(Sh_log).Cells(i_log, 2) = "ColName: "
        Sheets(Sh_log).Cells(i_log, 3) = Matrix_ColNameColIndex(i, 0)
        Sheets(Sh_log).Cells(i_log, 4) = "not found in " & Sh
        i_log = i_log + 1
        'The column existence check has detected an unfound column name
        z_ChkColExistence = False
    End If
Next i
End Function

Function MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from As String, ByRef
ColNames_from As Variant, _
    Sh_to As String, ByRef ColNames_to As Variant, _
    Optional KeyRow_from As Integer, Optional KeyRow_to As Integer) As Variant
'Determine the column indizes in Sh and Sh_new,
Dim ColName_from_i As String
Dim ColName_to_i As String
ReDim Matrix_ColName1Index1_ColName2Index2(0 To UBound(ColNames_from), 0 To 3) As
Variant

For i = LBound(ColNames_from) To UBound(ColNames_from) Step 1
    ColName_from_i = CStr(ColNames_from(i))
    Matrix_ColName1Index1_ColName2Index2(i, 0) = ColName_from_i
    Matrix_ColName1Index1_ColName2Index2(i, 1) = z_GetColumnIndex(ColName_from_i,
KeyRow_from, Sh_from)
    ColName_to_i = CStr(ColNames_to(i))
    Matrix_ColName1Index1_ColName2Index2(i, 2) = ColName_to_i
    Matrix_ColName1Index1_ColName2Index2(i, 3) = z_GetColumnIndex(ColName_to_i,
KeyRow_to, Sh_to)
    Debug.Print Matrix_ColName1Index1_ColName2Index2(i, 0) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 1) _
        & " " & Matrix_ColName1Index1_ColName2Index2(i, 2) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 3)
Next i
    MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex =
Matrix_ColName1Index1_ColName2Index2
End Function

Private Sub testMapFkt()
'Clear the sheet Log1
Sheets("Log1").Activate
Cells.Select
Selection.ClearContents

```

```

Stop
'Clear the yellow cells in Sheet1
Sheets("Sheet1").Activate
Range(Cells(2, 3), Cells(22, 14)).ClearContents
Stop
'Define the Attributes from the source sheet Sh_from=Sheets2
Dim ColNames_from As Variant
ColNames_from = Array("Attr1", "Attr3", "Attr2", "Attr7", _
    "Attr9", "Attr5", "Attr8", "Attr4", "Attr6")
'Define the attributes from the target sheet Sh_to=Sheets1
'Even though the attribute names in Sh_from and Sh_to may be called differently they must
'refer to the same attribute and the have to be in the same order in both arrays!!!
Dim ColNames_to As Variant
ColNames_to = Array("Attribute1", "Attribute3", "Attribute2", "Attribute7", _
    "Attribute9", "Attribute5", "Attribute8", "Attribute4", "Attribute6")
Call z_ShMapColumns("Sheet2", "Key1", ColNames_from, "Sheet1", "Key", ColNames_to, "Log1")
Stop
End Sub

```

```

*****
'customer%
*****

```

```

Function z_RenameCol(ColName_Old As String, ColName_New As String, Sh As String, Optional Row
As Long) As Integer

```

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

    Sheets(Sh).Activate
    If Row = Empty Then
        Row = 1
    End If
    On Error GoTo ColName_Old_NotFound
    Rows(Row).Find(What:=CStr(ColName_Old), LookAt:=xlWhole).Select
    On Error GoTo 0
    ActiveCell.Value = ColName_New
    z_RenameCol = True
    Exit Function
ColName_Old_NotFound:
    If z_ColExistent(ColName_New, Row, Sh) = True Then
        z_RenameCol = -1
    Else
        z_RenameCol = 0
    End If
End Function

```

```

Function z_ColExistent(ColName, Row, Sh) As Boolean

```

```

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

    On Error GoTo ColName_Old_NotFound
    Rows(Row).Find(What:=CStr(ColName), LookAt:=xlWhole).Select
    z_ColExistent = True
    On Error GoTo 0
    Exit Function
ColName_Old_NotFound:
    z_ColExistent = False

```

End Function

Function z\_ListOfSortedAttributeEntries(ColName As String, Sh\_from As String, Row\_to As Long, Col\_to As Long, Sh\_to As String)

'Author: Franz Schuermann, Project Management Excellence

'Roland Benz, Project Management Excellence

'Date: 27.9.2011

'Input: "RD\_MasterDataSet"

'Output: "RD\_MasterDataSet", "Logfile\_CustomerCheck"

'Objective:

'Create a logfile with all unique entries in the attribute "ListOfTaskCustomers"

'Correct the wrong entries in the column "ListOfTaskCustomers" (user interaction)

'Create a logfile of tasks with wrong entries in the attribute "ListOfTaskCustomers"

'Give the list to the PIMs (user interaction)

Dim ColIndex As Long

Dim RowSize As Long

ColIndex = z\_GetColumnIndex(ColName, 1, Sh\_from)

RowSize = z\_RowSize(1, Sh\_from)

'Copy the column from Sh\_from to Sh\_to

Call z\_CopyInsertRange(1, ColIndex, RowSize, ColIndex, Sh\_from, 1, 1, Sh\_to)

'call the filter and write out into (col\_to+1) all different entries (unique:=true) in column A

'Range(Cells(1, 1), Cells(RowSize, 1)).AdvancedFilter Action:=xlFilterCopy, CopyToRange:=Range( \_  
"B1"), Unique:=True

Sheets(Sh\_to).Range(Cells(Row\_to, Col\_to), Cells(RowSize, Col\_to)).AdvancedFilter \_  
Action:=xlFilterCopy, CopyToRange:=Cells(Row\_to, (Col\_to + 1)), Unique:=True

'define the sort range

'ActiveWorkbook.Worksheets(Sh\_To).Sort.SortFields.Add Key:=Range("B2:B100"), \_  
SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:= \_  
xlSortNormal

Dim RowSize\_To As Long

RowSize\_To = z\_RowSize(Col\_to + 1, Sh\_to)

ActiveWorkbook.Worksheets(Sh\_to).Sort.SortFields.Clear

ActiveWorkbook.Worksheets(Sh\_to).Sort.SortFields.Add Key:=Range(Cells(Row\_to + 1, Col\_to + 1),  
Cells(RowSize\_To, Col\_to + 1)), \_  
SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:= \_  
xlSortNormal

'apply the sort within the sort range

With ActiveWorkbook.Worksheets(Sh\_to).Sort

.SetRange Range(Cells(Row\_to + 1, Col\_to + 1), Cells(RowSize\_To, Col\_to + 1))

.Header = xlYes

.MatchCase = False

.Orientation = xlTopToBottom

.SortMethod = xlPinYin

.Apply

End With

'give a new column name for (Col\_to + 1)

Cells(1, Col\_to + 1).Value = "z\_ListOfSortedAttributeEntries: " & Cells(1, Col\_to + 1).Value  
Range("B1").EntireColumn.AutoFit

End Function

```

Function z_StringSplit(Dltrlst As Variant, Rpllst As Variant, RowU_From As Long, RowD_From As
Long, _
    Col_From As Long, Sh_from As String, RowU_To As Long, Col_to As Long, Sh_to As String, _
    Optional ByVal Wb As Workbook) As Variant
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'returns a matrix containing the customers and percents and write the result into Sh_log colC to
col...
    Sheets(Sh_from).Select
'copy the Range_from in Sh_from and paste it to Range_to in Sh_to
    Call z_CopyPasteRange(RowU_From, Col_From, RowD_From, Col_From, Sh_from, RowU_To,
Col_to, Sh_to)
'set up the range_to (even better if it were an input parameter of z_StringSplit)
    Sheets(Sh_to).Select
    Dim Rng As Range
    Set Rng = Range(Cells(RowU_To, Col_to), Cells(RowD_From, Col_to))
    Rng.Select
'Applied to the selected range. Replace in each string the delimiter list by the replacement list
    For k = LBound(Dltrlst) To UBound(Dltrlst)
        If Rpllst(k) = Empty Then
            I = Rpllst(0)
        Else
            I = k
        End If
        Selection.Replace What:=Dltrlst(k), Replacement:=Rpllst(I), LookAt:=xlPart, _
        SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
        ReplaceFormat:=False
    Next k
    Dim StringSplit_Dim1 As Variant
    Dim NofCustomers As Integer
'Applied to the selected range. Calculation of the nof customers by finding the character
combination "@@@"
    NofCustomers = z_NofCustomers(Rng, "@@@", Sh_to)
'declare the matrix StringSplit_Dim2 by using NofCustomers
    ReDim StringSplit_Dim2(RowU_From To RowD_From, Col_to + 1 To Col_to + 1 + (NofCustomers *
2)) As Variant
    For i = RowU_To To RowD_From
        'temporary split of one row
        StringSplit_Dim1 = Split(Cells(i, Col_to), "@")
        p = 0
        For m = LBound(StringSplit_Dim1) To UBound(StringSplit_Dim1)
            'fill the matrix StringSplit_Dim2 for each row i
            If StringSplit_Dim1(m) <> Empty Then
                Cells(i, Col_to + p + 1) = StringSplit_Dim1(m)
                StringSplit_Dim2(i, Col_to + p + 1) = StringSplit_Dim1(m)
                p = p + 1
            End If
        Next m
    Next i
'return the matrix
    z_StringSplit = StringSplit_Dim2
End Function

Function z_NofCustomers(Rng As Range, sLookupName As String, Sh As String) As Integer
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

'Calculation of the nof customers by finding the character combination sLookupName in a string
'The output is a
Dim sNames As String
Dim RngSize As Variant
'determine the size of the range (bug in not used output with index 4)
'RngSize(1)=nof rows
'RngSize(2)=nof cols
RngSize = z_RangeSize(Rng, Sh)
ReDim scount(RngSize(1)) As Variant
'Applied to the selected Range
For Row = 1 To RngSize(1) Step 1
    For Col = 1 To RngSize(2) Step 1
        'store the string in sNames
        sNames = Rng.Cells(Row, Col)
        'formula to determine how much sLookupName is found within the string (other possibility:
        UBound(Split(Str, a)))
        scount(Row) = (Len(sNames) - Len(Replace(sNames, sLookupName, ""))) / Len(sLookupName)
    Next Col
Next Row
'Give back the maximum value in array scount plus 1
z_NoCustomers = Application.WorksheetFunction.Max(scount) + 1
End Function

Function z_RangeSize(Rng As Range, Sh As String) As Variant
'output is an array(1 to 4)
'(1:nof selected rows, 2:nof selected cols, 3:index of last row in the range, 4:)
Sheets(Sh).Select
Rng.Select
Dim RangeSize(1 To 4) As Long
'nof selected rows
RangeSize(1) = Rng.Rows.Count
'nof selected columns
RangeSize(2) = Rng.Columns.Count
'index of last row in the range
RangeSize(3) = Rng.End(xlDown).Row
'index of the last col in the range (bug?)
RangeSize(4) = Rng.End(xlToRight).Column
z_RangeSize = RangeSize
End Function

Function z_GeneratePivotTable(Piv_ULCell As Range, Source_Rng As Range, _
    Sh_Source As String, Sh_Pivot As String, Piv_F_R_C_V As Variant) As String
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input parameters (arguments) for the z_GeneratePivotTable function and its call
'Name of the pivot sheet
'Sh_Pivot = "TimeToDateByTask"
'Name of the source sheet
'Sh_Source = "ActualsByWeek"
'Determine the range of Sh_Source and create a range object
'RowU = 1: ColL = 1
'RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
'Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'Create a range object for the upper left corner of the pivot
'Piv_sULCell = "B9"

```

```

        'Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
'Determine the pivot fields
        'Piv_F_R_C_V(0) = Array(13, 9, 11, 7, 12, 10) 'filter
        'Piv_F_R_C_V(1) = Array(17, 5) 'row labels
        'Piv_F_R_C_V(2) = Array() 'column labels
        'Piv_F_R_C_V(3) = Array(16) 'values

'Creat the strings for the function ActiveWorkbook.PivotCaches.Create() further below
Dim sSource_Rng As String
sSource_Rng = Sh_Source & "!" & Source_Rng.Address(ReferenceStyle:=xlR1C1)
Dim sPivot_Rng As String
sPivot_Rng = Sh_Pivot & "!" & Piv_ULCell.Address(ReferenceStyle:=xlR1C1)

'Store the range.address information into an array
Dim RngAddress As Variant
RngAddress = z_RangeAddressAsArray(Source_Rng)

'Store the Column names into an array
ReDim PivChosenField(RngAddress(2) To RngAddress(4)) As String
For i = RngAddress(2) To RngAddress(4)
    PivChosenField(i) = Source_Rng.Cells(1, i)
Next i

'Store the column names of the ReportFilter, RowLabel, ColLabel and Value into arrays
ReDim PivReportFilter(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivRowLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivColLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivValue(RngAddress(2) - 1 To RngAddress(4) - 1) As String
For i = RngAddress(2) - 1 To RngAddress(4) - 1
    On Error Resume Next
    PivReportFilter(i) = PivChosenField(Piv_F_R_C_V(0)(i))
    On Error GoTo 0
    On Error Resume Next
    PivRowLabel(i) = PivChosenField(Piv_F_R_C_V(1)(i))
    On Error GoTo 0
    On Error Resume Next
    PivColLabel(i) = PivChosenField(Piv_F_R_C_V(2)(i))
    On Error GoTo 0
    On Error Resume Next
    PivValue(i) = PivChosenField(Piv_F_R_C_V(3)(i))
    On Error GoTo 0
Next i

'generate Pivot
Sheets(Sh_Pivot).Select
Piv_ULCell.Select
ActiveWorkbook.PivotCaches.Create( _
    SourceType:=xlDatabase, _
    SourceData:=sSource_Rng, _
    Version:=xlPivotTableVersion12).CreatePivotTable _
    TableDestination:=sPivot_Rng, _
    TableName:=Piv_Name, _
    DefaultVersion:=xlPivotTableVersion12

```

```
'get Pivot table name
'if PivName = "PivotTable" is used more than once an iteger is added to the name
PivName = ActiveSheet.PivotTables(1).Name
```

```
For i = RngAddress(2) - 1 To RngAddress(4) - 1
    'Define row labels
    On Error Resume Next
    With ActiveSheet.PivotTables(PivName).PivotFields(PivRowLabel(i))
        .Orientation = xlRowField
        .Position = 1
    End With
    On Error GoTo 0
    'Define column labels
    On Error Resume Next
    With ActiveSheet.PivotTables(PivName).PivotFields(PivCollabel(i))
        .Orientation = xlColumnField
        .Position = 1
    End With
    On Error GoTo 0
    'Define values
    On Error Resume Next
    ActiveSheet.PivotTables(PivName).AddDataField ActiveSheet.PivotTables( _
    PivName).PivotFields(PivValue(i)), "Sum of STAFF_DAYS", xlSum
    On Error GoTo 0
    'Define report filters
    On Error Resume Next
    With ActiveSheet.PivotTables(PivName).PivotFields(PivReportFilter(i))
        .Orientation = xlPageField
        .Position = 1
    End With
    On Error GoTo 0
Next i
```

```
'Change the layout
With ActiveSheet.PivotTables(PivName)
    .InGridDropZones = True
    .RowAxisLayout xlTabularRow
End With
```

```
'Define all columns from : Choose fields to add to report
For i = RngAddress(2) To RngAddress(4)
    ActiveSheet.PivotTables(PivName).PivotFields(PivChosenField(i)).Subtotals = _
    Array(False, False, False, False, False, False, False, False, False, False, False, False)
Next i
```

```
'output
z_GeneratePivotTable = PivName
```

```
End Function
```

```
Function z_RangeAddressAsArray(Rng As Range) As Variant
```

```
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```



```

'Date: 26.9.2011
'Input: Range(Cells(a,b),Cells(c,d)), Output: Array(a,b,c,d)
sRngAddress = Rng.Address(ReferenceStyle:=xlR1C1)
Dltrlst = Array("$", "R", "C", ":")
Rpplst = Array("@", "@", "@", "@")
For k = LBound(Dltrlst) To UBound(Dltrlst)
    If Rpplst(k) = Empty Then
        l = Rpplst(0)
    Else
        l = k
    End If
    sRngAddress = Replace(sRngAddress, Dltrlst(k), Rpplst(l))
Next k
sRngAddress = Replace(sRngAddress, Rpplst(0) & Rpplst(0), Rpplst(l))
Dim RngAddress As Variant
RngAddress = Split(sRngAddress, Rpplst(0))
For i = 1 To 4 Step 1
    RngAddress(i - 1) = RngAddress(i)
Next i
ReDim Preserve RngAddress(1 To 4)
z_RangeAddressAsArray = RngAddress
End Function

```

```

Function z_RemWholeProject_WithAttrEntryOnPiLev(Sh As String, Sh_log As String, RefColName As String, _
    AttrColName As String, SearchAttrValue As String, Optional fkt_flag As Integer)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 12.10.2011

```

```

'Deletes all activities of a project if on Pi level the SyngentaPortfolioLevel1 entry is SEEDS
'Call z_RemWholeProject_WithAttrEntryOnPiLev("Sheet1", "Sh_log", _
'    "PiIdentifier", "SyngentaPortfolioLevel1", "SEEDS")

```

```

'Activate sheet and find column indices
Sheets(Sh).Activate
Dim Index_Ref As Long
Dim Index_Attr As Long
Index_Ref = z_GetColumnIndex(RefColName, 1, Sh)
Index_Attr = z_GetColumnIndex(AttrColName, 1, Sh)
'Look for the AttrValue and delete the entire project. Write out the row into Sh_log
Dim Row As Long: Row = 2
Dim ilog As Long: ilog = 2
Do Until Cells(Row, Index_Ref) = ""
    If fkt_flag = 0 Then
        AttrValue = Cells(Row, Index_Attr)
    End If
    'do not change the string function inside the UDF if necessary, but add new ones with new flags
    If fkt_flag = 1 Then
        AttrValue = left(Cells(Row, Index_Attr), 2) 'change the function if necessary or add others with
new flag
    End If
    'Delete the whole project if the condition on Pi Level is fulfilled
    '    attribute value on levels below Pi are not checked!

```

```

If AttrValue = SearchAttrValue Then
    Do Until Cells(Row, Index_Ref) <> Cells(Row + 1, Index_Ref)
        If 1 Then
            'Write out into the logfile
            Cells(Row, Index_Ref).EntireRow.Copy _
                Destination:=Sheets(Sh_log).Cells(ilog, 1).EntireRow: ilog = ilog + 1
        End If
        'Delete the entire row
        Cells(Row, Index_Ref).EntireRow.Delete Shift:=xlUp
        'Row = Row - 1
    Loop
    If 1 Then
        'Write out into the logfile
        Cells(Row, Index_Ref).EntireRow.Copy _
            Destination:=Sheets(Sh_log).Cells(ilog, 1).EntireRow: ilog = ilog + 1
    End If
    'Delete the entire row
    Cells(Row, Index_Ref).EntireRow.Delete Shift:=xlUp
    'Row = Row - 1
Else
    Do Until Cells(Row, Index_Ref) <> Cells(Row + 1, Index_Ref)
        Row = Row + 1
    Loop
    Row = Row + 1
End If
Loop
Sheets(Sh_log).Cells(1, 1) = "z_RemRow_WithAttrEntry"

End Function

Function z_RemRow_WithAttrEntry(Sh As String, Sh_log As String, RefColName As String, _
    AttrColName As String, SearchAttrValue As String, Optional fkt_flag As Integer)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 12.10.2011

'Removes the row if the ActivityIdentifier value is TK, RefColName is a key like Pild or ActivityId
'Call z_RemRow_WithAttrEntry("Sheet1", "Sh_log", _
'    "Pildentifier", "ActivityIdentifier", "TK", 1)

'Activate sheet and find column indices
Sheets(Sh).Activate
Dim Index_Ref As Long
Dim Index_Attr As Long
Index_Ref = z_GetColumnIndex(RefColName, 1, Sh)
Index_Attr = z_GetColumnIndex(AttrColName, 1, Sh)
'Look for the AttrValue and delete the entire row. Write out the row into Sh_log
Dim AttrValue As Variant
Dim Row As Long: Row = 2
Dim ilog As Long: ilog = 2
Do Until Cells(Row, Index_Ref) = ""
    If fkt_flag = 0 Then
        AttrValue = Cells(Row, Index_Attr)
    End If

```

```

'do not change the string function inside the UDF if necessary, but add new ones with new flags
If fkt_flag = 1 Then
    AttrValue = left(Cells(Row, Index_Attr), 2)
End If
If AttrValue = SearchAttrValue Then
    If 1 Then
        'Write out into the logfile
        Cells(Row, Index_Ref).EntireRow.Copy _
            Destination:=Sheets(Sh_log).Cells(iLog, 1).EntireRow: iLog = iLog + 1
    End If
    'Delete the entire row
    Cells(Row, Index_Ref).EntireRow.Delete Shift:=xlUp
    Row = Row - 1
End If
Row = Row + 1
Loop
Sheets(Sh_log).Cells(1, 1) = "z_RemRow_WithAttrEntry"
End Function

```

```

Function z_AutoFilterOn(Sh As String, Row As Long)
    'Autofilter on
    Sheets(Sh).Select
    Rows(Row).Select
    If Sheets(Sh).AutoFilterMode = False Then
        Selection.AutoFilter
    End If
    'show all data
    If Sheets(Sh).FilterMode = True Then
        ActiveSheet.ShowAllData
    End If
End Function

```

```

Function z_AutoFilterOff(Sh As String, Row As Long)
    'Autofilter on
    Sheets(Sh).Select
    Rows(Row).Select
    If Sheets(Sh).AutoFilterMode = True Then
        Selection.AutoFilter
    End If
End Function

```

```

Function TextFile_Create(ByRef txt_file As Object, PathAndName As String)
    'call
    'Call TextFile_Create(txt_file, "C:\Users\t740698\Desktop\testfile.txt")
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set txt_file = fs.CreateTextFile(PathAndName, True)
End Function

```

```

Function TextFile_Open_Or_CreateAndOpen(ByRef txt_file As Object, PathAndName As String)
    'Call:
    'Call TextFile_Open_Or_CreateAndOpen(txt_file, "C:\Users\t740698\Desktop\testfile.txt")
    'if an error message pops up then the file is already open
    Set fs = CreateObject("Scripting.FileSystemObject")
    '8=For Appending at the end of the file

```

```

'True=Create a new file if it does not exist
Set txt_file = fs.OpenTextFile(PathAndName, 8, True)
End Function
Function TextFile_WriteInto1(txt_file As Object, txt_arr() As Variant)
'Call:
'Dim txt(0 to 2) As Variant
'txt(0) = "Start - Task1: " & CStr(Now())
'txt(1) = "Stop - Task1: " & CStr(Now())
'txt(2) = "Start - Task2: " & CStr(Now())
'Call TextFile_WriteInto1(txt_file, txt)
For i = LBound(txt_arr) To UBound(txt_arr)
    txt_file.WriteLine (txt_arr(i))
Next i
End Function
Function TextFile_WriteInto2(txt_file As Object, txt_arr As Variant)
'Call:
'Call TextFile_WriteInto2(txt_file, Array("string1 " & CStr(Now()), "string2 " & CStr(Now())))
For i = LBound(txt_arr) To UBound(txt_arr)
    txt_file.WriteLine (txt_arr(i))
Next i
End Function
Function TextFile_Close(txt_file As Object)
    txt_file.Close
End Function

Function z_WriteAttributeEntriesIntoCellsBelow(Sh As String, Optional Rng As Range)
    Sheets(Sh).Activate
    Dim ColSize As Long
    Dim RowSize As Long
    If Not Rng Is Nothing Then
        'Store the range.address information into an array
        Dim Rng_Address As Variant
        Rng_Address = z_RangeAddressToArray(Rng)
        RowSize = Rng_Address(3)
        ColSize = Rng_Address(4)
    Else
        ColSize = z_ColSize(1, Sh)
        RowSize = z_LastWrittenRow(Sh, 1, ColSize)
    End If

    Dim Col As Long
    Dim Row As Long
    Col = 1
    Row = 2
    Do While Col < ColSize + 1
        Do While Row < RowSize + 1
            Cells(Row, Col).Select
            Dim Value_Cell_store As Variant
            If Cells(Row, Col).Value <> "" Then 'Empty="" or 0 but ""<>0
                'store the value
                Value_Cell_store = Cells(Row, Col).Value
                'move down
                Row = Row + 1
            End If
            Col = Col + 1
        Loop
        Row = Row + 1
    Loop
End Function

```

```

        Cells(Row, Col).Select
    End If
    'move the rows down and write
    'If Row = 498 Then Stop
    Do While Cells(Row, Col).Value = "" 'Empty="" or 0 but ""<>0
        'termination criteria
        If Row = RowSize Then
            If Cells(Row, 1).Value Like "**Total*" Then
                'Col = Col + 1
                'Row = 2
                Row = Row + 1
                Exit Do
            End If
        End If
        If Row > RowSize Then
            'Col = Col + 1
            'Row = 2
            Cells(Row, Col).Select
            If Col > ColSize Then
                Exit Function
            End If
            Exit Do
        End If
        'write
        Cells(Row, Col).Value = Value_Cell_store
        'move down
        Row = Row + 1
        Cells(Row, Col).Select
    Loop
    Loop
    Col = Col + 1
    Row = 2
    Loop
End Function
Sub test134245()
    FirstRow = z_Rng_firstRow(Range(Cells(2, 3), Cells(5, 10)))
    lastRow = z_Rng_lastRow(Range(Cells(2, 3), Cells(5, 10)))
    firstCol = z_Rng_firstCol(Range(Cells(2, 3), Cells(5, 10)))
    lastCol = z_Rng_lastCol(Range(Cells(2, 3), Cells(5, 10)))
    CountRow = z_Rng_countRow(Range(Cells(2, 3), Cells(5, 10)))
    CountCol = z_Rng_countCol(Range(Cells(2, 3), Cells(5, 10)))
    Value = z_Rng_Key_Value(Range(Cells(2, 3), Cells(5, 10)))
End Sub
Function z_Rng_firstRow(ByRef Rng As Range) As Long
    z_Rng_firstRow = Rng(1, 1).Row
End Function
Function z_Rng_lastRow(ByRef Rng As Range) As Long
    Dim Rng_firstRow As Long
    Rng_firstRow = Rng(1, 1).Row
    z_Rng_lastRow = Rng_firstRow + Rng.Rows.Count - 1
End Function
Function z_Rng_firstCol(ByRef Rng As Range) As Long
    z_Rng_firstCol = Rng(1, 1).Column

```

```

End Function
Function z_Rng_lastCol(ByRef Rng As Range) As Long
    Dim Rng_firstCol As Long
    Rng_firstCol = Rng(1, 1).Column
    z_Rng_lastCol = Rng_firstCol + Rng.Columns.Count - 1
End Function
Function z_Rng_countRow(ByRef Rng As Range) As Long
    z_Rng_countRow = Rng.Rows.Count
End Function
Function z_Rng_countCol(ByRef Rng As Range) As Long
    z_Rng_countCol = Rng.Columns.Count
End Function
Function z_Rng_Key_Value(ByRef Rng As Range) As Variant
    z_Rng_Key_Value = Rng(1, 1).Value
End Function

```

```

Sub z_MapAttributes_MouseClickVersion()
    'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim Input_Range_Key_from As Range
    Dim Input_Range_Attributes_from As Range
    Dim Input_Range_Key_to As Range
    Dim Input_Range_Attributes_to As Range
    Dim Input_Indices_Key_from As Variant
    Dim Input_Indices_Attributes_from As Variant
    Dim Input_Indices_Key_to As Variant
    Dim Input_Indices_Attributes_to As Variant

    Dim Sh_from As String
    Dim Sh_to As String

    'inputs
    On Error GoTo EXITSUB
    Set Input_Range_Key_from = Application.InputBox(prompt:="Select the Key_from and click OK",
Type:=8)
    Input_Indices_Key_from = z_RangeToIndices(Input_Range_Key_from)

    Sh_from = Input_Range_Key_from.Worksheet.Name
    Sheets(Sh_from).Activate

    Set Input_Range_Attributes_from = Application.InputBox(prompt:="Select the Attribute_from and
click OK", Type:=8)
    Input_Indices_Attributes_from = z_RangeToIndices(Input_Range_Attributes_from)

    Set Input_Range_Key_to = Application.InputBox(prompt:="Select the Key_to and click OK",
Type:=8)
    Input_Indices_Key_to = z_RangeToIndices(Input_Range_Key_to)

    Sh_to = Input_Range_Key_to.Worksheet.Name
    Sheets(Sh_to).Activate

```

```

Set Input_Range_Attributes_to = Application.InputBox(prompt:="Select the Attribute_to and click OK", Type:=8)
Input_Indices_Attributes_to = z_RangeToIndices(Input_Range_Attributes_to)
On Error GoTo 0

```

```

Dim Key_from As String
ReDim ColNames_from(0 To Input_Indices_Attributes_from(3) - Input_Indices_Attributes_from(1)) As Variant
Dim Key_to As String
ReDim ColNames_to(0 To Input_Indices_Attributes_to(3) - Input_Indices_Attributes_to(1)) As Variant

```

```

Response = MsgBox("Sh_from= " & CStr(Sh_from) & Chr(13) & _
    "Key_from= " & CStr(Input_Range_Key_from) & Chr(13) & _
    "Attribute_from= " & CStr(Input_Range_Attributes_from(1, 1)) & " : " & _
    CStr(Input_Range_Attributes_from.End(xlToRight).Value) & Chr(13) & _
    "Sh_to= " & CStr(Sh_to) & Chr(13) & _
    "Key_to= " & CStr(Input_Range_Key_to) & Chr(13) & _
    "Attribute_to= " & CStr(Input_Range_Attributes_to(1, 1)) & " : " & _
    CStr(Input_Range_Attributes_to.End(xlToRight).Value) & Chr(13), vbYesNoCancel)

```

```

If Response = vbCancel Then
    Exit Sub
ElseIf Response = vbNo Then
    Exit Sub
ElseIf Response = vbYes Then

```

```

End If

```

```

Key_from = Input_Range_Key_from.Value
Key_to = Input_Range_Key_to.Value

```

```

For iter = 0 To Input_Indices_Attributes_from(3) - Input_Indices_Attributes_from(1)
    ColNames_from(iter) = Input_Range_Attributes_from(1, iter + 1).Value
'Array(Input_Range_Attributes_from.Value)
    ColNames_to(iter) = Input_Range_Attributes_to(1, iter + 1).Value
'Array(Input_Range_Attributes_to.Value)
Next iter

```

```

'Function z_ShMapColumns(Sh_from As String, ColName_Key_from As String, ByRef
ColNames_from As Variant, _
'    Sh_to As String, ColName_Key_to As String, ByRef ColNames_to As Variant, _
'    Optional Sh_log As String, Optional ByRef Wb As Workbook, _
'    Optional KeyRow_from As Integer, Optional KeyRow_to As Integer)
Call z_ShMapColumns(Sh_from, Key_from, ColNames_from, Sh_to, Key_to, ColNames_to)
Exit Sub
EXITSUB:
Exit Sub
End Sub

```

```

*****

```

```

'Input via Mouse clicks

```

\*\*\*\*\*

Private Function z\_RangeToIndices(ByRef Rng As Range) As Variant

'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

Dim RangeIndices(0 To 3) As Long

Dim CellsArray() As String

Dim sAddr As String

sAddr = Rng.Address(ReferenceStyle:=xlR1C1)

CellsArray = Split(sAddr, ":")

Dim CellIndicesUL() As Long

On Error GoTo RangelsColumnOrRow

CellIndicesUL = z\_sCellToIndex(CellsArray(0))

On Error GoTo 0

Dim CellIndicesLR() As Long

On Error GoTo RangelsCell

CellIndicesLR = z\_sCellToIndex(CellsArray(1))

On Error GoTo 0

RangeIndices(0) = CellIndicesUL(0)

RangeIndices(1) = CellIndicesUL(1)

RangeIndices(2) = CellIndicesLR(0)

RangeIndices(3) = CellIndicesLR(1)

z\_RangeToIndices = RangeIndices

Exit Function

RangelsCell:

CellIndicesLR = z\_sCellToIndex(CellsArray(0))

Resume Next

RangelsColumnOrRow:

Dim RorC As String

RorC = left(sAddr, 1)

OneOrMore = InStr(1, sAddr, ":", vbTextCompare)

'only one row or column

If OneOrMore = 0 Then

    If RorC = "C" Then

        RangeIndices(0) = 1

        RangeIndices(1) = z\_sColumnToIndex(CellsArray(0))

        RangeIndices(2) = 1048534

        RangeIndices(3) = RangeIndices(0)

    Elseif RorC = "R" Then

        RangeIndices(0) = z\_sRowToIndex(CellsArray(0))

        RangeIndices(1) = 1

        RangeIndices(2) = RangeIndices(0)

        RangeIndices(3) = 16383

    Else

        Stop

    End If

'more than one row or column

Else

    If RorC = "C" Then

        RangeIndices(0) = 1



```

        RangeIndices(1) = z_sColumnToIndex(CellsArray(0))
        RangeIndices(2) = 1048534
        RangeIndices(3) = z_sColumnToIndex(CellsArray(1))
    ElseIf RorC = "R" Then
        RangeIndices(0) = z_sRowToIndex(CellsArray(0))
        RangeIndices(1) = 1
        RangeIndices(2) = z_sRowToIndex(CellsArray(1))
        RangeIndices(3) = 16383
    Else
        Stop
    End If
End If
z_RangeToIndices = RangeIndices
End Function

```

```

Private Function z_sCellToIndex(ByRef CellIndexStr As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim CellIndexArr() As String 'Splited R1C1 Format
    Dim ColIndex As Long
    Dim RowIndex As Long
    Dim CellIndices(0 To 1) As Long
    'find column index
    CellIndexArr = Split(CellIndexStr, "C")
    ColIndex = CLng(CellIndexArr(1))
    CellIndexArr = Split(CellIndexArr(0), "R")
    RowIndex = CLng(CellIndexArr(1))
    CellIndices(0) = RowIndex
    CellIndices(1) = ColIndex
    'Output
    z_sCellToIndex = CellIndices
End Function

```

```

Private Function z_sColumnToIndex(ByRef ColIndexStrLeft As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim ColArray() As String
    ColArray = Split(ColIndexStrLeft, "C")
    z_sColumnToIndex = ColArray(1)
End Function

```

```

Private Function z_sRowToIndex(ByRef RowIndexStrUp As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim RowArray() As String
    RowArray = Split(RowIndexStrUp, "R")
    z_sRowToIndex = RowArray(1)
End Function

```

```

*****
'Checks whether ColName exists
*****

```

```

Private Function z_ChkColExistence(Sh As String, ByRef ColNames As Variant, Sh_log As String) As Boolean
    'The column existence check is assumed to find all column names at the beginning

```

```

z_ChkColExistence = True
'Determine the column indices of the ColNames array in Sh
Dim ColName_i As String
ReDim Matrix_ColNameColIndex(0 To UBound(ColNames), 0 To 1) As Variant
'iterate through the array
For i = LBound(ColNames_from) To UBound(ColNames) Step 1
    ColName_i = CStr(ColNames(i))
    Matrix_ColNameColIndex(i, 0) = ColName_i
    Matrix_ColNameColIndex(i, 1) = z_GetColumnIndex(ColName_i, 1, Sh)
    Debug.Print Matrix_ColNameColIndex(i, 0) & " " & Matrix_ColNameColIndex(i, 1)
    If Matrix_ColNameColIndex(i, 1) = 0 Then
        'write errors into the logfile
        Sheets(Sh_log).Cells(iLog, 2) = "ColName: "
        Sheets(Sh_log).Cells(iLog, 3) = Matrix_ColNameColIndex(i, 0)
        Sheets(Sh_log).Cells(iLog, 4) = "not found in " & Sh
        iLog = iLog + 1
        'The column existence check has detected an unfound column name
        z_ChkColExistence = False
    End If
Next i
End Function

*****

'Mapping
*****

Private Function z_ShMapColumns(Sh_from As String, ColName_Key_from As String, ByRef
ColNames_from As Variant, _
    Sh_to As String, ColName_Key_to As String, ByRef ColNames_to As Variant, _
    Optional Sh_log As String, Optional ByRef Wb As Workbook, _
    Optional KeyRow_from As Integer, Optional KeyRow_to As Integer)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Application.ScreenUpdating = False
'Start time measuring
Dim Start As Date: Dim Duration As Date
Start = Now()

'optional KeyRows
If KeyRow_from = 0 Then
    KeyRow_from = 1
End If
If KeyRow_to = 0 Then
    KeyRow_to = 1
End If

'create a matrix with column names and indexes
MapMatrix = MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from,
ColNames_from, Sh_to, ColNames_to, _
    KeyRow_from, KeyRow_to)

'Find the column index of the KeyName in "Sh_from"
Dim ColIndex_Key_from As Long

```

```

ColIndex_Key_from = z_GetColumnIndex(ColName_Key_from, KeyRow_from, Sh_from)

'Find the column index of the KeyName in "Sh_to"
Dim ColIndex_Key_to As Long
ColIndex_Key_to = z_GetColumnIndex(ColName_Key_to, KeyRow_to, Sh_to)

'Determine the row size in Sh_to
Sheets(Sh_to).Activate
RowSize_To = z_RowSize(ColIndex_Key_to, Sh_to)

'Select the range in the column Key_to
Sheets(Sh_to).Activate
Range(Cells(2, ColIndex_Key_to), Cells(RowSize_To, ColIndex_Key_to)).Select

'Iterate through the rows with "rcheck" = PIdentifier
Dim ilog As Long
ilog = 2
For Each ValueInCol_Key_to In Selection.Cells
    'if ValueInCol_Key_to is found in Sh_from then perform the mapping
    If Not Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to,
LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns with the help of the MapMatrix
        For j = LBound(MapMatrix) To UBound(MapMatrix) Step 1
            'read the indices
            ColIndex_from_j = MapMatrix(j, 1)
            ColIndex_to_j = MapMatrix(j, 3)
            'map
            ValueInCol_Key_to.Offset(0, (ColIndex_to_j) - ColIndex_Key_to).Value = _
                Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to, _
                LookAt:=xlWhole).Offset(0, (ColIndex_from_j) - ColIndex_Key_from)
        Next j
    Else
        'write not found ValueInCol_Key_to in Sh_from into Sh_log
        'Sh_log is optional, if not existent as input resume next
        On Error Resume Next
        Sheets(Sh_log).Cells(ilog, 2) = ValueInCol_Key_to
        Sheets(Sh_log).Cells(ilog, 4) = " not found, map them from another source file Sh_from"
        ilog = ilog + 1
        On Error GoTo 0
    End If
Next

'In case the mapping has changed the row height
Sheets(Sh_to).Activate
Cells.Select
Selection.Rows.RowHeight = 15
Application.ScreenUpdating = True
'Write the durations into the logfile
'Sh_log is optional, if not existent as input resume next
Duration = Now() - Start
On Error Resume Next
Sheets(Sh_log).Cells(ilog + 2, 1) = "Duration" & CStr(Duration)
On Error GoTo 0

```

End Function

```
Private Function MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from As String,
ByRef ColNames_from As Variant, _
    Sh_to As String, ByRef ColNames_to As Variant, _
    Optional KeyRow_from As Integer, Optional KeyRow_to As Integer) As Variant
'Determine the column indizes in Sh and Sh_new,
Dim ColName_from_i As String
Dim ColName_to_i As String
ReDim Matrix_ColName1Index1_ColName2Index2(0 To UBound(ColNames_from), 0 To 3) As
Variant

For i = LBound(ColNames_from) To UBound(ColNames_from) Step 1
    ColName_from_i = CStr(ColNames_from(i))
    Matrix_ColName1Index1_ColName2Index2(i, 0) = ColName_from_i
    Matrix_ColName1Index1_ColName2Index2(i, 1) = z_GetColumnIndex(ColName_from_i,
KeyRow_from, Sh_from)
    ColName_to_i = CStr(ColNames_to(i))
    Matrix_ColName1Index1_ColName2Index2(i, 2) = ColName_to_i
    Matrix_ColName1Index1_ColName2Index2(i, 3) = z_GetColumnIndex(ColName_to_i,
KeyRow_to, Sh_to)
    Debug.Print Matrix_ColName1Index1_ColName2Index2(i, 0) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 1) _
        & " " & Matrix_ColName1Index1_ColName2Index2(i, 2) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 3)
Next i
    MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex =
Matrix_ColName1Index1_ColName2Index2
End Function
```

```
Private Function z_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, _
    Optional Sh As String, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Output datatype change from Variant
Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer
'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
Sheets(Sh).Activate
'find column name
On Error GoTo NameExpectedNotExist:
Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole).Select
On Error GoTo 0
'find column index
CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
'Output
z_GetColumnIndex = ColIndex
```

```

Exit Function
OptionalArgument:
Resume Next
NameExpectedNotExistent:
CollIndex = 0
z_GetColumnIndex = CollIndex
End Function

```

```

Private Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As
Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet
Sheets(Sh).Activate
'Determine the row size
z_RowSize = If(IsEmpty(Cells(1048576, SearchCol)), Cells(1048576, SearchCol).End(xlUp).Row,
1048576)
End Function

```

```

Sub m_VLookUp()
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Dim KeyRange_from As Range
Dim AttributesRange_from As Range
Dim KeyRange_to As Range
Dim AttributesRange_to As Range
Dim KeyAddress_from As String
Dim AttributesAddress_from As String
Dim KeyAddress_to As String
Dim AttributesAddress_to As String

Dim Sh_from As String
Dim Sh_to As String

'inputs
On Error GoTo EXITSUB

'Key from
Set KeyRange_from = Application.InputBox(prompt:="Select the Key_from and click OK", Type:=8)
KeyAddress_from = KeyRange_from.Address
'Sheet from
Sh_from = KeyRange_from.Worksheet.Name
Sheets(Sh_from).Activate
'Attributes from
Set AttributesRange_from = Application.InputBox(prompt:="Select the Attributes_from and click
OK", Type:=8)
AttributesAddress_from = AttributesRange_from.Address

'Key to
Set KeyRange_to = Application.InputBox(prompt:="Select the Key_to and click OK", Type:=8)
KeyAddress_to = KeyRange_to.Address

```

```

'Sheet to
Sh_to = KeyRange_to.Worksheet.Name
Sheets(Sh_to).Activate
'Attributes to
Set AttributesRange_to = Application.InputBox(prompt:="Select the Attributes_to and click OK",
Type:=8)
AttributesAddress_to = AttributesRange_to.Address

On Error GoTo 0

' Dim Key_from As String
' ReDim ColNames_from(0 To Input_Indices_Attributes_from(3) -
Input_Indices_Attributes_from(1)) As Variant
' Dim Key_to As String
' ReDim ColNames_to(0 To Input_Indices_Attributes_to(3) - Input_Indices_Attributes_to(1)) As
Variant
'
' Response = MsgBox("Sh_from= " & CStr(Sh_From) & Chr(13) & _
' "Key_from= " & CStr(Input_Range_Key_from) & Chr(13) & _
' "Attribute_from= " & CStr(Input_Range_Attributes_from(1, 1)) & " : " &
CStr(Input_Range_Attributes_from.End(xlToRight).Value) & Chr(13) & _
' "Sh_to= " & CStr(Sh_To) & Chr(13) & _
' "Key_to= " & CStr(Input_Range_Key_to) & Chr(13) & _
' "Attribute_to= " & CStr(Input_Range_Attributes_to(1, 1)) & " : " &
CStr(Input_Range_Attributes_to.End(xlToRight).Value) & Chr(13), vbYesNoCancel)
'
' If Response = vbCancel Then
' Exit Sub
' ElseIf Response = vbNo Then
' Exit Sub
' ElseIf Response = vbYes Then
'
' End If

'log
Dim Sh_log As String
On Error Resume Next
Sh_log = Sheets("Log").Name
On Error GoTo 0

'mapping
Call z_MapAttributeRanges(Sh_from, KeyRange_from, AttributesRange_from, _
Sh_to, KeyRange_to, AttributesRange_to, Sh_log)

Exit Sub
EXITSUB:
Exit Sub
End Sub

Private Function z_MapAttributeRanges(Sh_from As String, KeyRange_from As Range,
AttributesRange_from As Range, _
Sh_to As String, KeyRange_to As Range, AttributesRange_to As Range, _

```

Optional Sh\_log As String, Optional ByRef Wb As Workbook)

```
'Stop
'get the indices
'-----
'only one Key from
If KeyRange_from.Count <> 1 Then
    MsgBox "error"
    Stop
    Exit Function
End If
Dim Row_Key_from As Long
Dim Col_Key_from As Long
Row_Key_from = KeyRange_from.Row
Col_Key_from = KeyRange_from.Column

'only one Key to
If KeyRange_to.Count <> 1 Then
    MsgBox "error"
    Stop
    Exit Function
End If
Dim Row_Key_to As Long
Dim Col_Key_to As Long
Row_Key_to = KeyRange_to.Row
Col_Key_to = KeyRange_to.Column

'Attribute from
Dim cnt As Long
Dim Item_ As Range
cnt = 0
For Each Item_ In AttributesRange_from.Areas
    For iter = 0 To Item_.Count - 1
        cnt = cnt + 1
    Next
Next
ReDim Row_AttributesRange_from(0 To cnt - 1) As Variant
ReDim Col_AttributesRange_from(0 To cnt - 1) As Variant
cnt = 0
For Each Item_ In AttributesRange_from.Areas
    For iter = 0 To Item_.Count - 1
        Row_AttributesRange_from(cnt) = Item_.Offset(0, iter).Row
        Col_AttributesRange_from(cnt) = Item_.Offset(0, iter).Column
        cnt = cnt + 1
    Next
Next

'Attributes to
cnt = 0
For Each Item_ In AttributesRange_to.Areas
    For iter = 0 To Item_.Count - 1
        cnt = cnt + 1
    Next
```

```

Next
ReDim Row_AttributesRange_to(0 To cnt - 1) As Variant
ReDim Col_AttributesRange_to(0 To cnt - 1) As Variant
cnt = 0
For Each Item_ In AttributesRange_to.Areas
    For iter = 0 To Item_.Count - 1
        Row_AttributesRange_to(cnt) = Item_.Offset(0, iter).Row
        Col_AttributesRange_to(cnt) = Item_.Offset(0, iter).Column
        cnt = cnt + 1
    Next
Next

'tests
'-----
'the same nof attributes
If UBound(Row_AttributesRange_from) <> UBound(Row_AttributesRange_to) Then
    MsgBox "error"
    Stop
    Exit Function
End If
'everything on one line
For iter = 0 To UBound(Row_AttributesRange_from)
    If Row_AttributesRange_from(iter) <> Row_Key_from Then
        MsgBox "error"
        Stop
        Exit Function
    End If
    If Row_AttributesRange_to(iter) <> Row_Key_to Then
        MsgBox "error"
        Stop
        Exit Function
    End If
Next

'mapping matrix
'-----
'create a matrix with column names and indexes
MapMatrix = Make_MapMatrix(Col_Key_from, Col_AttributesRange_from, Col_Key_to,
Col_AttributesRange_to)

'mapping
'-----
'Determine the row size in Sh_to
Sheets(Sh_from).Activate
RowSize_from = z_RowSize(Col_Key_from, Sh_from)

'Select the range in the column Key_to
Sheets(Sh_from).Activate
Range(Cells(Row_Key_from + 1, Col_Key_from), Cells(RowSize_from, Col_Key_from)).Select

'test whether there are key_from.values
If Row_Key_from = RowSize_from Then
    MsgBox "error"

```



```

    Stop
    Exit Function
End If

'Iterate through the rows
Dim Col_from_j As Long
Dim Col_to_j As Long
Dim ilog As Long
ilog = 2

On Error Resume Next
Exists = (Sheets(Sh_log).Name <> "")
On Error GoTo 0
If Exists = True Then
    For Each CellInCol_Key_from In Selection.Cells
        'if ValueInCol_Key_to is found in Sh_from then perform the mapping
        If Not Sheets(Sh_to).Columns(Col_Key_to).Find(What:=CellInCol_Key_from.Value,
LookAt:=xlWhole) Is Nothing Then
            'iterate through the columns with the help of the MapMatrix
            For j = LBound(MapMatrix) To UBound(MapMatrix) Step 1
                'read the indices
                Col_from_j = MapMatrix(j, 0)
                Col_to_j = MapMatrix(j, 1)
                'map
                Sheets(Sh_to).Columns(Col_Key_to).Find(What:=CellInCol_Key_from.Value, _
                LookAt:=xlWhole).Offset(0, (Col_to_j) - Col_Key_to) = _
                CellInCol_Key_from.Offset(0, Col_from_j - Col_Key_from).Value
            Next j
        Else
            'write not found ValueInCol_Key_to in Sh_from into Sh_log
            'Sh_log is optional, if not existent as input resume next
            On Error Resume Next
            Sheets(Sh_log).Cells(ilog, 2) = CellInCol_Key_from.Value
            Sheets(Sh_log).Cells(ilog, 4) = " Key_from.value not found in Key_to.value"
            ilog = ilog + 1
            On Error GoTo 0
        End If
    Next
End If

'Determine the row size in Sh_to
Sheets(Sh_to).Activate
RowSize_To = z_RowSize(Col_Key_to, Sh_to)

'Select the range in the column Key_to
Sheets(Sh_to).Activate
Range(Cells(Row_Key_to + 1, Col_Key_to), Cells(RowSize_To, Col_Key_to)).Select

For Each CellInCol_Key_to In Selection.Cells
    'if ValueInCol_Key_to is found in Sh_from then perform the mapping
    If Not Sheets(Sh_from).Columns(Col_Key_from).Find(What:=CellInCol_Key_to.Value,
LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns with the help of the MapMatrix

```

```

For j = LBound(MapMatrix) To UBound(MapMatrix) Step 1
    'read the indices
    ColIndex_from_j = MapMatrix(j, 0)
    ColIndex_to_j = MapMatrix(j, 1)
    'map
    Sheets(Sh_from).Columns(Col_Key_from).Find(What:=CellInCol_Key_to.Value, _
    '    LookAt:=xlWhole).Offset(0, (Col_from_j) - Col_Key_from) = _
    '    CellInCol_Key_to.Offset(0, Col_to_j - Col_Key_to).Value
    CellInCol_Key_to.Offset(0, (ColIndex_to_j) - Col_Key_to).Value = _
    Sheets(Sh_from).Columns(Col_Key_from).Find(What:=CellInCol_Key_to.Value, _
    '    LookAt:=xlWhole).Offset(0, (ColIndex_from_j) - Col_Key_from)
Next j
Else
    'write not found ValueInCol_Key_to in Sh_from into Sh_log
    'Sh_log is optional, if not existent as input resume next
    On Error Resume Next
    Sheets(Sh_log).Cells(ilog, 2) = CellInCol_Key_to.Value
    Sheets(Sh_log).Cells(ilog, 4) = " Key_to.value not found in Key_from.value"
    ilog = ilog + 1
    On Error GoTo 0
End If
Next

'In case the mapping has changed the row height
Sheets(Sh_to).Activate
Cells.Select
Selection.Rows.RowHeight = 15
Application.ScreenUpdating = True

'Stop
End Function

Private Function Make_MapMatrix(Col_Key_from As Long, Col_AttributesRange_from As Variant, _
    Col_Key_to As Long, Col_AttributesRange_to As Variant) As Variant

    ReDim Matrix(0 To UBound(Col_AttributesRange_from), 0 To 1) As Variant

    For i = LBound(Col_AttributesRange_from) To UBound(Col_AttributesRange_from) Step 1
        Matrix(i, 0) = Col_AttributesRange_from(i)
        Matrix(i, 1) = Col_AttributesRange_to(i)
    Next i
    Make_MapMatrix = Matrix
End Function

Private Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As
Workbook) As Long
    'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    'Date: 26.9.2011
    'Input: column, Output: row with the last entry in that column
    'SearchCol datatype changed from integer
    'Activate the Sheet
    Sheets(Sh).Activate
    'Determine the row size

```

```

    z_RowSize = If(IsEmpty(Cells(1048576, SearchCol)), Cells(1048576, SearchCol).End(xlUp).Row,
1048576)
End Function

```

```

Sub m_PasteValues()
'
' m_PasteValues Macro
'
' Keyboard Shortcut: Ctrl+Shift+V
'
    Selection.PasteSpecial Paste:=xlPasteValuesAndNumberFormats, Operation:= _
        xlNone, SkipBlanks:=False, Transpose:=False
    Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False
    Selection.PasteSpecial Paste:=xlPasteColumnWidths, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False
End Sub

```

```

Sub m_PasteFormats()
'
' m_PasteFormats Macro
'
' Keyboard Shortcut: Ctrl+Shift+F
'
    Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False
End Sub

```

```

Sub m_FindNotExistentPIs()
    Dim Sh As String
    Dim SearchColFrom As Long
    Dim SearchColIn As Long
    Sh = Application.InputBox("Enter the sheet name as string")
    SearchColFrom = Application.InputBox("Enter column of where to search from as long")
    SearchColIn = Application.InputBox("Enter column of where to search in as long")
    Call z_FindNotExistentPIs(Sh, SearchColFrom, SearchColIn)
    'Call z_FindNotExistentPIs("CP - EAC Full Costs", 1, 11)
End Sub

```

```

Function z_FindNotExistentPIs(Sh As String, SearchColFrom As Long, SearchColIn As Long)
    Dim rngSearchIn As Range
    Dim rngFound As Range
    Sheets(Sh).Activate
    RowSizeIn = If(IsEmpty(Cells(1048576, SearchColIn)), Cells(1048576, SearchColIn).End(xlUp).Row,
1048576)
    RowSizeWhat = If(IsEmpty(Cells(1048576, SearchColFrom)), Cells(1048576,
SearchColFrom).End(xlUp).Row, 1048576)
    If RowSizeIn > RowSizeWhat Then
        RowSize = RowSizeIn
    Else
        RowSize = RowSizeWhat
    End If
End Function

```

```

End If
Set rngSearchIn = Range(Cells(2, SearchColIn), Cells(RowSize, SearchColIn))
' Loop through it
For Row = 2 To RowSize
    'get the search text
    SearchText = Cells(Row, SearchColFrom)
    ' Find our required text
    Set rngFound = rngSearchIn.Find(SearchText)
    ' If we find it then...
    If Not rngFound Is Nothing Then
        Cells(Row, SearchColFrom).Font.ColorIndex = 5
        rngFound.Font.ColorIndex = 5
    'if we do not find it then...
    Else
        Cells(Row, SearchColFrom).Font.ColorIndex = 3
    End If
Next
End Function

```

```

Private Sub ResourceEACsToPiEACs()
    Call z_ResourceEACsToPiEACs("PMEC_VBA_MasterDataSet_12", "Comparison_EACs_")
End Sub
Function z_ResourceEACsToPiEACs(Sh As String, Sh_log As String)
    ReDim Pild_Arr(0 To 10000) As Variant
    Dim ColName_i As String
    Dim Pild_Col As Long
    Dim EAC_col_i As Long
    Dim Pi_i As String
    Dim EAC_Sum As Double
    Dim RowLog As Long
    Dim Rng_Pi_i As Range
    'fill the array with all PIs
    Pild_Col = z_GetColumnIndex("Pildentifier", 1, Sh)
    RowSize = z_RowSize(Pild_Col, Sh)
    iter = 0
    For Row = 2 To RowSize
        If iter = 0 Then
            Pild_Arr(iter) = Cells(Row, Pild_Col)
            iter = iter + 1
        ElseIf Pild_Arr(iter - 1) <> Cells(Row, 1) Then
            Pild_Arr(iter) = Cells(Row, Pild_Col)
            iter = iter + 1
        End If
    Next Row
    'Redim the array, remove the empty indexes
    For iter = 0 To UBound(Pild_Arr)
        If Pild_Arr(iter) = Empty Then
            ReDim Preserve Pild_Arr(0 To iter - 1)
            Exit For
        End If
    Next iter

```

```

'calculate the EAC for all PIs
ColName_i = "EAC Full Costs_c"
EAC_col_i = z_GetColumnIndex(ColName_i, 1, Sh)
RowLog = 2
For Pilter = LBound(Pild_Arr) To UBound(Pild_Arr)
    'calculate EAC_sum for one Pi (Pi_i)
    Pi_i = Pild_Arr(Pilter)
    Call z_SelectMultiple2(Sh, Pild_Col, Pi_i, Rng_Pi_i)
    EAC_Sum = WorksheetFunction.Sum(Rng_Pi_i.Columns(EAC_col_i))
    'write out the result
    Sheets(Sh_log).Cells(RowLog, 1) = Pi_i
    Sheets(Sh_log).Cells(RowLog, 2) = EAC_Sum
    RowLog = RowLog + 1
Next Pilter
End Function

```

#### Option Compare Text

```

Sub m_GeneratePivotTable()
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Name of the source sheet
Dim Sh_Source As String
Sh_Source = ActiveSheet.Name
If Sh_Source = "Pivot" Then
    MsgBox ("Select the sheet with the source data and restart the macro.")
    Exit Sub
End If
'Name of the pivot sheet
Dim Sh_Pivot As String
Dim Exists As Boolean
Exists = False
Sh_Pivot = "Pivot"
Dim WSh As Worksheet
For Each WSh In ActiveWorkbook.Worksheets
    If WSh.Name = Sh_Pivot Then
        'Clear
        flag_no_endlessloop = True 'does not work!?!
        WSh.Select
        Cells.Select
        Application.EnableEvents = False
        Selection.Delete
        Application.EnableEvents = True
        Exists = True
        Exit For
    End If
Next
If Exists = False Then
    Call z_ShNew(Sh_Pivot, "Before", Sheets(Sh_Source))
End If
'Determine the range of Sh_Source and create a range object
'variant 1
'RowU = 1: ColL = 1

```

```

'RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
'Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'variant 2
'Set Source_Rng = Sheets(Sh_Source).UsedRange
'variant 3
Dim ColSize As Long
Dim RowSize As Long
Dim Source_Rng As Range
Sheets(Sh_Source).Activate
ColSize = z_ColSize(1, Sh_Source)
RowSize = z_LastWrittenRow(Sh_Source, 1, ColSize)
Set Source_Rng = Sheets(Sh_Source).Range(Cells(1, 1), Cells(RowSize, ColSize))
'Create a range object for the upper left corner of the pivot
Dim Piv_sULCell As String
Dim Piv_ULCell As Range
Piv_sULCell = "B4"
Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
'generate pivot
Call z_GeneratePivotTable(Sh_Source, Source_Rng, Sh_Pivot, Piv_ULCell)
End Sub

Private Function z_GeneratePivotTable(Sh_Source As String, Source_Rng As Range, _
    Sh_Pivot As String, Piv_ULCell As Range) As String
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input parameters (arguments) for the z_GeneratePivotTable function and its call
'Name of the pivot sheet
'Sh_Pivot = "TimeToDateByTask"
'Name of the source sheet
'Sh_Source = "ActualsByWeek"
'Determine the range of Sh_Source and create a range object
'RowU = 1: ColL = 1
'RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
'Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'Create a range object for the upper left corner of the pivot
'Piv_sULCell = "B9"
'Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)

'Create the strings for the function ActiveWorkbook.PivotCaches.Create() further below
Dim sSource_Rng As String
sSource_Rng = Sh_Source & "!" & Source_Rng.Address(ReferenceStyle:=xlR1C1)
Dim sPivot_Rng As String
sPivot_Rng = Sh_Pivot & "!" & Piv_ULCell.Address(ReferenceStyle:=xlR1C1)

'Store the range.address information into an array
Dim RngAddress As Variant
RngAddress = z_RangeAddressAsArray(Source_Rng)

'Store the Column names into an array
ReDim PivChosenField(RngAddress(2) To RngAddress(4)) As String
For i = RngAddress(2) To RngAddress(4)
    PivChosenField(i) = Source_Rng.Cells(1, i)

```

Next i

'Turn off events

Application.EnableEvents = False

'generate Pivot

Sheets(Sh\_Pivot).Select

Piv\_ULCell.Select

ActiveWorkbook.PivotCaches.Create( \_

    SourceType:=xlDatabase, \_

    SourceData:=sSource\_Rng, \_

    Version:=xlPivotTableVersion12).CreatePivotTable \_

    TableDestination:=sPivot\_Rng, \_

    TableName:=Piv\_Name, \_

    DefaultVersion:=xlPivotTableVersion12

'get Pivot table name

PivName = ActiveSheet.PivotTables(1).Name

Dim pt As PivotTable

Dim pf As PivotField

Set pt = Sheets("Pivot").PivotTables(PivName)

pt.ManualUpdate = True

'find Attributes like

Dim Identifier\_ As String

Dim Title\_ As String

Dim Value\_ As String

For i = RngAddress(2) To RngAddress(4)

    Dim test\_1 As Boolean

    Dim test\_2 As Boolean

    test\_1 = PivChosenField(i) Like "Pi\*Id\*"

    test\_2 = Not PivChosenField(i) Like "\*Tit\*"

    If test\_1 = True And test\_2 = True Then

        Identifier\_ = PivChosenField(i)

        Exit For

    End If

Next i

For i = RngAddress(2) To RngAddress(4)

    test\_1 = PivChosenField(i) Like "Pi\*Title\*"

    test\_2 = Not PivChosenField(i) Like "\*Id\*"

    If test\_1 = True And test\_2 = True Then

        Title\_ = PivChosenField(i)

        Exit For

    End If

Next i

For i = RngAddress(2) To RngAddress(4)

    test\_1 = False

    test\_1 = PivChosenField(i) Like "\*Eac?Full\*12\*"

    If test\_1 = True Then

        Value\_ = PivChosenField(i)

```

        Exit For
    End If
Next i

'find identifier
On Error Resume Next
With ActiveSheet.PivotTables(PivName).PivotFields(Identifier_)
    .Orientation = xlRowField
    .Position = 1
End With
On Error GoTo 0
'find title
On Error Resume Next
With ActiveSheet.PivotTables(PivName).PivotFields(Title_)
    .Orientation = xlRowField
    .Position = 2
End With
On Error GoTo 0
'find cost
'Define values
On Error Resume Next
ActiveSheet.PivotTables(PivName).AddDataField ActiveSheet.PivotTables( _
    PivName).PivotFields(Value_), , xlSum
On Error GoTo 0

'Change the layout in: tabular form and not Classic layout
With ActiveSheet.PivotTables(PivName)
    .InGridDropZones = False
    .RowAxisLayout xlTabularRow
End With

'Define all columns from : Remove Subtotals
' For i = RngAddress(2) To RngAddress(4)
'     ActiveSheet.PivotTables(PivName).PivotFields(PivChosenField(i)).Subtotals = _
'     Array(False, False, False, False, False, False, False, False, False, False, False, False)
' Next i
For Each pf In pt.PivotFields
    pf.Subtotals = _
        Array(False, False, False, False, False, False, False, False, False, False, False, False)
Next pf

'set datafield settings
For Each pf In pt.DataFields
    pf.Function = xlSum
    pf.NumberFormat = "#,##0.00_);[Red](#,##0.00)"
Next pf

pt.ManualUpdate = False

'colwidth
Sheets(Sh_Pivot).Activate
'Range(Cells(1, 1), Cells(1, 20)).EntireColumn.AutoFit 'this line takes hours to run!!!
For Col = 1 To 20

```



```

    If Cells(1, Col).ColumnWidth > 30 Then
        Cells(1, Col).ColumnWidth = 30
    End If
Next Col

```

```

'Turn on events
Application.EnableEvents = True

```

```

'add reference Extensibility 5.3
Call z_SetReferenceExtensibility5_3

```

```

'generate module and code
Call z_AddMacros

```

```

'call one of the generated macros
Call z_RunMacroWithEarlyBinding

```

```

'output
z_GeneratePivotTable = PivName

```

End Function

```

Private Sub z_AddMacros()
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

Dim ProcKind As VBIDE.vbext_ProcKind
Dim LineNum As Long
Dim CodeName_ As String
Dim Name As String
Dim sCode As String
Dim VBComp_Exists As Boolean
VBComp_Exists = False

```

```

'Module "GlobalVariables", and macro SetFlag and Global variable flag_no_endless_loop
sCode = "Public flag_no_endlessloop As Boolean" & vbCrLf & vbCrLf & _
    "Sub SetFlag()" & vbCrLf & _
    "    flag_no_endlessloop = false" & vbCrLf & _
    "End Sub"

```

```

For Each VBComp In ActiveWorkbook.VBProject.VBComponents
    If VBComp.Name = "GlobalVariables" Then
        VBComp_Exists = True
        With ActiveWorkbook.VBProject.VBComponents("GlobalVariables").CodeModule
            LineNum = .CountOfDeclarationLines + 1
            ProcName = .ProcOfLine(LineNum, ProcKind)
        End With
        If ProcName = Empty Then
            Call z_AddCodeToModule("GlobalVariables", sCode)
        Else
            'Do nothing otherwise Excel crashes
        End If
    Exit For
End If
Next

```

```

If VBComp_Exists = False Then
    Call z_AddModuleAndCode("GlobalVariables", sCode)
End If

```

```

'Module "ThisWorkbook", Macro Workbook_Open
sCode = "Private Sub Workbook_Open()" & vbCrLf & _
    "    flag_no_endlessloop = False" & vbCrLf & _
    "End Sub"
With ActiveWorkbook.VBProject.VBComponents("ThisWorkbook").CodeModule
    LineNum = .CountOfDeclarationLines + 1
    ProcName = .ProcOfLine(LineNum, ProcKind)
End With
If ProcName = Empty Then
    Call z_AddCodeToModule("ThisWorkbook", sCode)
Else
    'Do nothing otherwise Excel crashes
End If

```

```

'Module "Pivot", Macro Worksheet_Change
sCode = "Private Sub Worksheet_Change(ByVal Target As Range)" & vbCrLf
sCode = sCode & "    If flag_no_endlessloop = True Then" & vbCrLf
sCode = sCode & "        Exit Sub" & vbCrLf
sCode = sCode & "    End If" & vbCrLf
sCode = sCode & "    Dim pt As PivotTable" & vbCrLf
sCode = sCode & "    Dim pf As PivotField" & vbCrLf
sCode = sCode & "    Set pt = ActiveSheet.PivotTables(1)" & vbCrLf
sCode = sCode & "    pt.ManualUpdate = True" & vbCrLf
sCode = sCode & "    flag_no_endlessloop = True" & vbCrLf
sCode = sCode & "    Application.EnableEvents = False" & vbCrLf
sCode = sCode & "    For Each pf In pt.DataFields" & vbCrLf
sCode = sCode & "        If Not pf Is Nothing Then" & vbCrLf
sCode = sCode & "            With pf" & vbCrLf
sCode = sCode & "                .Function = xlSum" & vbCrLf
sCode = sCode & "                .NumberFormat = ""#,##0.00_);[Red](#,##0.00)"" & vbCrLf
sCode = sCode & "            End With" & vbCrLf
sCode = sCode & "        End If" & vbCrLf
sCode = sCode & "    Next pf" & vbCrLf
sCode = sCode & "    pt.ManualUpdate = False" & vbCrLf
sCode = sCode & "    ActiveSheet.Activate" & vbCrLf
'sCode = sCode & "    Range(Cells(1, 1), Cells(1, 20)).EntireColumn.AutoFit" & vbCrLf
sCode = sCode & "    For Col = 1 To 20" & vbCrLf
sCode = sCode & "        If Cells(1, Col).ColumnWidth > 30 Then" & vbCrLf
sCode = sCode & "            Cells(1, Col).ColumnWidth = 30" & vbCrLf
sCode = sCode & "        End If" & vbCrLf
sCode = sCode & "    Next Col" & vbCrLf
sCode = sCode & "    flag_no_endlessloop = False" & vbCrLf
sCode = sCode & "    Application.EnableEvents = True" & vbCrLf
sCode = sCode & "End Sub" & vbCrLf
CodeName_ = Sheets("Pivot").CodeName
With ActiveWorkbook.VBProject.VBComponents(CodeName_).CodeModule
    LineNum = .CountOfDeclarationLines + 1
    ProcName = .ProcOfLine(LineNum, ProcKind)
End With

```

```

If ProcName = Empty Then
    Call z_AddCodeToModule(CodeName_, sCode)
Else
    'Do nothing otherwise Excel crashes
End If

```

```

' With ActiveWorkbook.VBProject.VBComponents(CodeName_).CodeModule
' Dim LineNum As Long
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "Private Sub Worksheet_Change(ByVal Target As Range)"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    If flag_no_endlessloop = True Then"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "        Exit Sub"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    End If"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    Dim pt As PivotTable"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    Dim pf As PivotField"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    Set pt = Sheets("""Pivot""").PivotTables(1)"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    pt.ManualUpdate = True"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    flag_no_endlessloop = True"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    For Each pf In pt.DataFields"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "        If Not pf Is Nothing Then"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "            With pf"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "                .Function = xlSum"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "                .NumberFormat = ""#,##0.00_);[Red](#,##0.00)""
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "            End With"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "        End If"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    Next pf"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    pt.ManualUpdate = False"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    Sheets("""Pivot""").Activate"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    Range(Cells(1, 1), Cells(1, 20)).EntireColumn.AutoFit"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "    For Col = 1 To 20"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "        If Cells(1, Col).ColumnWidth > 30 Then"
' LineNum = .CountOfLines + 1

```

```

' .InsertLines LineNum, "      Cells(1, Col).ColumnWidth = 30"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "      End If"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "      Next Col"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "      flag_no_endlessloop = False"
' LineNum = .CountOfLines + 1
' .InsertLines LineNum, "End Sub"
' End With

```

End Sub

```

Private Function z_AddModuleAndCode(sModulName As String, sCode As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

```

```

Dim Module_tmp As String
On Error Resume Next
If ActiveWorkbook.VBProject.VBComponents(sModulName) Is Nothing Then
'add moduleX
Module_tmp = ActiveWorkbook.VBProject.VBComponents.Add(vbext_ct_StdModule).Name
'rename moduleX
ActiveWorkbook.VBProject.VBComponents(Module_tmp).Name = sModulName
'add code
Call z_AddCodeToModule(sModulName, sCode)
Else
'delete code
Call z_RemoveCodeFromModule(sModulName)
'add code
Call z_AddCodeToModule(sModulName, sCode)
End If

```

End Function

```

Private Function z_AddCodeToModule(sModulName As String, sCode As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
ActiveWorkbook.VBProject.VBComponents(sModulName).CodeModule.AddFromString sCode
End Function

```

```

Private Function z_RemoveCodeFromModule(sModulName As String)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
With ActiveWorkbook.VBProject.VBComponents(sModulName).CodeModule
.DeleteLines 1, .CountOfLines
End With
End Function

```

```

Private Function z_RunMacroWithEarlyBinding()
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Call ActiveWorkbook.SetFlag 'does not exists because it is in another Wb
'Dim sFile As String
'Dim wkb As Workbook
'sFile = "vb07_test.xls"
'On Error Resume Next

```

```

'Set wkb = Workbooks(sFile)
'On Error GoTo 0
'If wkb Is Nothing Then
'  MsgBox "Die Testarbeitsmappe " & sFile & " wurde nicht gefunden!"
'Else
'  Run sFile & "!Tabelle1.CallClassModule"
'End If
'Replace line if it is a function to call
  'Run(sFile & "!CallerName", Application.Caller)

Application.Run "" & ActiveWorkbook.FullName & "!SetFlag"
End Function

Private Function z_SetReferenceExtensibility5_3()
On Error Resume Next
ThisWorkbook.VBProject.References.AddFromGuid _
  GUID="{0002E157-0000-0000-C000-000000000046}", _
  Major:=5, Minor:=3
On Error GoTo 0
End Function

Sub m_Pivot_To_FlatTable()
'Name of the source sheet
Dim Sh_Pivot As String
Dim WSh_Pivot As Worksheet
Sh_Pivot = ActiveSheet.Name
If Sh_Pivot <> "Pivot" Then
  MsgBox ("Select the sheet with the pivot table and restart the macro.")
  Exit Sub
Else
  Set WSh_Pivot = Sheets(Sh_Pivot)
End If

Dim pt As PivotTable
Dim Nof_Pt As Integer
Nof_Pt = 0
For Each pt In WSh_Pivot.PivotTables
  Nof_Pt = Nof_Pt + 1
Next pt
If Nof_Pt <> 1 Then
  MsgBox ("Make sure you habe only one Pivot in the sheet.")
  Exit Sub
Else
  Set pt = WSh_Pivot.PivotTables(1)
End If

Call z_ShNew("Pivot_Flat", "Before:=", WSh_Pivot)
Dim Rng_pt As Range
Set Rng_pt = pt.TableRange1
WSh_Pivot.Activate
Call z_CopyRange("Values", Sh_Pivot, Rng_pt, "Pivot_Flat", Sheets("Pivot_Flat").Range("A1"))

If Range("A1").Value = Empty Then

```

```

    Rows(1).Delete
    Dim Row_plus As Long
    Row_plus = 1
End If

'only if a value field
Dim ColSize As Long
Dim RowSize As Long
ColSize = z_ColSize(1, "Pivot_Flat")
RowSize = z_LastWrittenRow("Pivot_Flat", 1, ColSize)

Dim Rng_pt_Data As Range
Dim Row_pt_Data As Long
Dim Col_pt_Data As Long
Set Rng_pt_Data = pt.DataBodyRange
Row_pt_Data = Rng_pt_Data(1, 1).Row
Col_pt_Data = Rng_pt_Data(1, 1).Column

Dim Row_pt As Long
Dim Col_pt As Long
Row_pt = Rng_pt(1, 1).Row
Col_pt = Rng_pt(1, 1).Column

Dim Row_Off As Long
Dim Col_Off As Long
Row_Off = Row_pt - 1
Col_Off = Col_pt - 1
'subtotals
If Cells(RowSize, 1).Value Like "*Total*" Then
    For Col = 2 To ColSize
        Sheets("Pivot_Flat").Cells(RowSize, Col).Select
        If Not Intersect(Sheets("Pivot").Cells(RowSize, Col). _
            Offset(Row_Off, Col_Off), Rng_pt_Data) Is Nothing Then
            Sheets("Pivot_Flat").Select
            Dim Rng_Subtotal As Range
            Set Rng_Subtotal = Range(Cells(2, Col), Cells(RowSize - 1, Col))
            Call z_InsertFormula_Subtotal("Pivot_Flat", Rng_Subtotal,
Sheets("Pivot_Flat").Cells(RowSize, Col))
        End If
    Next Col
End If
'zero instead of blank
For Col = 2 To ColSize
    If Not Intersect(Sheets("Pivot").Cells(RowSize, Col). _
        Offset(Row_Off, Col_Off), Rng_pt_Data) Is Nothing Then
        Cells(1, Col).Select
        Selection.EntireColumn.NumberFormat = "#,##0.00"
        For Row = 2 To RowSize
            If Cells(Row, Col).Value = "" Then
                Cells(Row, Col).Value = 0
            End If
        Next Row
        Range(Cells(2, Col), Cells(RowSize, Col)).Activate
    End If
Next Col

```

```

        'Selection.Replace What:="", Replacement:=0, LookAt:=xlPart
        Selection.Replace What:="", Replacement:=0, LookAt:=xlWhole
    End If
Next Col

Dim Rng_pt_Data_flat As Range
Set Rng_pt_Data_flat = Range(Cells(1, 1), Cells(RowSize, Col_pt_Data - 1 - Col_pt + 1))
Call z_WriteAttributeEntriesIntoCellsBelow("Pivot_Flat", Rng_pt_Data_flat)
Call z_AutofilterOn("Pivot_Flat", 1)
End Sub
Sub m_WriteAttributeEntriesIntoCellsBelow()
    'Select Range
    Dim Rng As Range
    Set Rng = Application.InputBox(prompt:="Select the Range and click OK", Type:=8)
    Address = Rng.Address
    'Sheet from
    Dim Sh As String
    Sh = Rng.Worksheet.Name
    Sheets(Sh).Activate
    Call z_WriteAttributeEntriesIntoCellsBelow(Sh, Rng)
End Sub
Private Function z_WriteAttributeEntriesIntoCellsBelow(Sh As String, Optional Rng As Range)
    Sheets(Sh).Activate
    Dim ColSize As Long
    Dim RowSize As Long
    If Not Rng Is Nothing Then
        'Store the range.address information into an array
        Dim Rng_Address As Variant
        Rng_Address = z_RangeAddressAsArray(Rng)
        RowSize = Rng_Address(3)
        ColSize = Rng_Address(4)
    Else
        ColSize = z_ColSize(1, Sh)
        RowSize = z_LastWrittenRow(Sh, 1, ColSize)
    End If

    Dim Col As Long
    Dim Row As Long
    Col = 1
    Row = 2
    Do While Col < ColSize + 1
        Do While Row < RowSize + 1
            Cells(Row, Col).Select
            Dim Value_Cell_store As Variant
            If Cells(Row, Col).Value <> "" Then 'Empty="" or 0 but ""<>0
                'store the value
                Value_Cell_store = Cells(Row, Col).Value
                'move down
                Row = Row + 1
                Cells(Row, Col).Select
            End If
            'move the rows down and write
            'If Row = 498 Then Stop

```

```

Do While Cells(Row, Col).Value = "" 'Empty="" or 0 but ""<>0
'termination criteria
If Row = RowSize Then
    If Cells(Row, 1).Value Like "*Total*" Then
        'Col = Col + 1
        'Row = 2
        Row = Row + 1
        Exit Do
    End If
End If
If Row > RowSize Then
    'Col = Col + 1
    'Row = 2
    Cells(Row, Col).Select
    If Col > ColSize Then
        Exit Function
    End If
    Exit Do
End If
'write
Cells(Row, Col).Value = Value_Cell_store
'move down
Row = Row + 1
Cells(Row, Col).Select
Loop
Loop
Col = Col + 1
Row = 2
Loop
End Function

```

```

Function z_InsertFormula_Subtotal(Sh As String, Rng_from As Range, Rng_to As Range)
    Sheets(Sh).Activate
    Dim Rng_AddressString As String
    Rng_AddressString = Rng_from.Address
    myformula = "=SUBTOTAL(109" & "," & Rng_AddressString & ")"
    Rng_to.Formula = myformula
End Function

```

\*\*\*\*\*

'Auxiliary functions

\*\*\*\*\*

```

Private Function z_ShNew(Sh As String, Optional Where As String, Optional ByRef Sh_Ref As
Worksheet, Optional ByRef Wb As Workbook)
    'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    'Date: 26.9.2011
    'Input: Name of the new Sh, where to place the new Sh, before or after some Sh_Ref, at the begin
or the end
    On Error Resume Next
    WorksheetExists = (Sheets(Sh).Name <> "")
    On Error GoTo 0
    If WorksheetExists = False Then
        Call z_ShAdd(Where, Sh_Ref)
    End If
End Function

```



```

    ActiveSheet.Name = Sh 'Worksheets.Add(Before:=Worksheets(1)).Name = Sh
End If
'Clear contents
Sheets(Sh).Activate
ActiveSheet.Cells.Select
Selection.ClearContents
'Format
Sheets(Sh).Columns.ColumnWidth = 20
Sheets(Sh).Rows.RowHeight = 15
End Function

```

```

Private Function z_ShAdd(Optional Where As String, Optional ByRef Sh_Ref As Worksheet)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: where to place the added Sh, before or after some Sh_Ref, at the begin or the end
If Not Sh_Ref Is Nothing Then
    If Where = ("Before:") Then
        Sheets.Add Sh_Ref
    ElseIf Where = ("After:") Then
        Sheets.Add , Sh_Ref
    Else
        Sheets.Add Before:=Sheets(1)
    End If
Else
    If Where = ("End") Then
        Sheets.Add After:=Sheets(Sheets.Count)
    ElseIf Where = ("Begin") Then
        Sheets.Add Before:=Sheets(1)
    Else
        Sheets.Add Before:=Sheets(1)
    End If
End If
End Function

```

```

Private Function z_ShDelete(Sh As String, Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
Wb.Activate
Application.DisplayAlerts = False
On Error Resume Next
Sheets(Sh).Delete
On Error GoTo 0
Application.DisplayAlerts = True
End Function

```

```

Private Function z_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef Wb As
Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Row, Output: column with the last entry in that row
'SearchCol datatype changed from integer
'Activate the Sheet
'Sheets(Sh).Activate

```

```

'Determine the col size
z_ColSize = If(IsEmpty(Sheets(Sh).Cells(SearchRow, 16384)), Sheets(Sh).Cells(SearchRow,
16384).End(xlToLeft).Column, 16384)
End Function

```

```

Private Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As
Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet
'Sheets(Sh).Activate
'Determine the row size
z_RowSize = If(IsEmpty(Sheets(Sh).Cells(1048576, SearchCol)), Sheets(Sh).Cells(1048576,
SearchCol).End(xlUp).Row, 1048576)
End Function

```

```

Private Function z_LastWrittenRow(Optional Sh As String, Optional StartAtCol As Long, _
Optional StopAtCol As Long, Optional ByRef Wb As Workbook) As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 11.10.2011
'Input: All Columns, Output: last written row
'FirstEmptyCol = z_FirstEmptyCol()
Dim RowSize_max As Long: RowSize_max = 0
Dim RowSize_col As Long: RowSize_col = 0
Dim AllCols As Long: AllCols = 16384
Dim Col As Long
'Optional input
If StartAtCol = 0 Then
    StopAtCol = 1
End If
If StopAtCol = 0 Then
    StopAtCol = AllCols
End If
For Col = StartAtCol To StopAtCol
    RowSize_col = z_RowSize(Col, Sh)
    If RowSize_col > RowSize_max Then
        RowSize_max = RowSize_col
    End If
Next Col
z_LastWrittenRow = RowSize_max
End Function

```

```

Private Function z_RangeAddressAsArray(Rng As Range) As Variant
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Range(Cells(a,b),Cells(c,d)), Output: Array(a,b,c,d)
sRngAddress = Rng.Address(ReferenceStyle:=xlR1C1)
Dltrlst = Array("$", "R", "C", ":")
Rpllst = Array("@", "@", "@", "@")
For k = LBound(Dltrlst) To UBound(Dltrlst)
    If Rpllst(k) = Empty Then

```

```

        l = RplLst(0)
    Else
        l = k
    End If
    sRngAddress = Replace(sRngAddress, Dltrlst(k), RplLst(l))
Next k
sRngAddress = Replace(sRngAddress, RplLst(0) & RplLst(0), RplLst(l))
Dim RngAddress As Variant
RngAddress = Split(sRngAddress, RplLst(0))
For i = 1 To 4 Step 1
    RngAddress(i - 1) = RngAddress(i)
Next i
ReDim Preserve RngAddress(1 To 4)
z_RangeAddressAsArray = RngAddress
End Function

```

```

Private Function z_AutofilterOn(Sh As String, Row As Long)
    'Autofilter on
    Sheets(Sh).Select
    Rows(Row).Select
    If Sheets(Sh).AutoFilterMode = False Then
        Selection.AutoFilter
    End If
    'show all data
    If Sheets(Sh).FilterMode = True Then
        ActiveSheet.ShowAllData
    End If
End Function

```

```

Private Function z_AutofilterOff(Sh As String, Row As Long)
    'Autofilter on
    Sheets(Sh).Select
    Rows(Row).Select
    If Sheets(Sh).AutoFilterMode = True Then
        Selection.AutoFilter
    End If
End Function

```

```

Private Function z_CopyRange(PasteAllOrValuesOrFormats As String, Sh_from As String, Range_From
As Range, _
    Sh_to As String, Cell_to As Range)
    'only works out if you invoke this function after this line!:Sheets(Sh_from).Activate
    'otherwise VBA cannot set the Range_From
    'set Cell_to as follows: Cell_to=Sheets(Sh_to).Range("A1")
    Sheets(Sh_from).Activate
    Range_From.Select
    Selection.Copy
    Sheets(Sh_to).Activate
    Cell_to.Select
    Dim first As Long
    Dim last As Long
    If PasteAllOrValuesOrFormats = "All" Then
        first = Range_From.Columns.End(xlToLeft).Column

```

```

        last = Range_From.Columns.End(xlToRight).Column
        Call z_Copy_ColWidth(Sh_from, Sh_to, first, last)
        Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    ElseIf PasteAllOrValuesOrFormats = "Values" Then
        Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    ElseIf PasteAllOrValuesOrFormats = "Formats" Then
        Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False
    End If
End Function

Private Function z_Copy_ColWidth(Sh_from As String, Sh_to As String, Col_From, Col_to)
    For Col = Col_From To Col_to
        Sheets(Sh_to).Columns(Col).ColumnWidth = Sheets(Sh_from).Columns(Col).ColumnWidth
    Next Col
End Function

```

```

*****
*****

```

```

Sub m_AddAllNotEacToRowLabels_PildFirst()
    'Name of the source sheet
    Dim Sh_Pivot As String
    Dim WSh_Pivot As Worksheet
    Sh_Pivot = ActiveSheet.Name
    If Sh_Pivot <> "Pivot" Then
        MsgBox ("Select the sheet with the pivot table and restart the macro.")
        Exit Sub
    Else
        Set WSh_Pivot = Sheets(Sh_Pivot)
    End If
    'pivot table
    Dim pt As PivotTable
    Dim Nof_Pt As Integer
    Nof_Pt = 0
    For Each pt In WSh_Pivot.PivotTables
        Nof_Pt = Nof_Pt + 1
    Next pt
    If Nof_Pt <> 1 Then
        MsgBox ("Make sure you habe only one Pivot in the sheet.")
        Exit Sub
    Else
        Set pt = WSh_Pivot.PivotTables(1)
    End If
    'Turn off events
    Application.EnableEvents = False
    'data fields
    Dim pf As PivotField
    Dim Pos As Long
    Pos = 1
    For Each pf In pt.PivotFields
        If pf.Name Like "Pi*Id*" And Not pf.Name Like "*Tit*" _

```

```

        Or pf.Name Like "PI*Id*" And Not pf.Name Like "**Tit*" Then
        pf.Orientation = xlRowField
    End If
    If Not pf.Name Like "*Eac*" Then
        pf.Orientation = xlRowField
    End If
Next
'Turn on events
Application.EnableEvents = True
End Sub

Sub m_AddAllEacToValues_PildFirst()
    'Name of the source sheet
    Dim Sh_Pivot As String
    Dim WSh_Pivot As Worksheet
    Sh_Pivot = ActiveSheet.Name
    If Sh_Pivot <> "Pivot" Then
        MsgBox ("Select the sheet with the pivot table and restart the macro.")
        Exit Sub
    Else
        Set WSh_Pivot = Sheets(Sh_Pivot)
    End If
    'pivot table
    Dim pt As PivotTable
    Dim Nof_Pt As Integer
    Nof_Pt = 0
    For Each pt In WSh_Pivot.PivotTables
        Nof_Pt = Nof_Pt + 1
    Next pt
    If Nof_Pt <> 1 Then
        MsgBox ("Make sure you habe only one Pivot in the sheet.")
        Exit Sub
    Else
        Set pt = WSh_Pivot.PivotTables(1)
    End If
    'Turn off events
    Application.EnableEvents = False
    'data fields
    Dim pf As PivotField
    Dim Pos As Long
    Pos = 1
    For Each pf In pt.PivotFields
        If pf.Name Like "Pi*Id*" And Not pf.Name Like "**Tit*" _
        Or pf.Name Like "PI*Id*" And Not pf.Name Like "**Tit*" Then
            pf.Orientation = xlRowField
        End If
        If pf.Name Like "*Eac*" Or pf.Name Like "**EAC*" Then
            pt.AddDataField pf, , xlSum
        End If
    Next
    'Turn on events
    Application.EnableEvents = True
End Sub

```

```

Sub m_RemoveAllNotEacFromRowLabels_PiIdRemains()
    'Name of the source sheet
    Dim Sh_Pivot As String
    Dim WSh_Pivot As Worksheet
    Sh_Pivot = ActiveSheet.Name
    If Sh_Pivot <> "Pivot" Then
        MsgBox ("Select the sheet with the pivot table and restart the macro.")
        Exit Sub
    Else
        Set WSh_Pivot = Sheets(Sh_Pivot)
    End If
    'pivot table
    Dim pt As PivotTable
    Dim Nof_Pt As Integer
    Nof_Pt = 0
    For Each pt In WSh_Pivot.PivotTables
        Nof_Pt = Nof_Pt + 1
    Next pt
    If Nof_Pt <> 1 Then
        MsgBox ("Make sure you habe only one Pivot in the sheet.")
        Exit Sub
    Else
        Set pt = WSh_Pivot.PivotTables(1)
    End If
    'Turn off events
    Application.EnableEvents = False
    'data fields
    Dim pf As PivotField
    Dim Pos As Long
    Pos = 1
    For Each pf In pt.PivotFields
        If Not pf.Name Like "*Eac*" And Not pf.Name Like "*EAC*" And Not pf.Name = "Values" Then
            If Not pf.Name Like "Pi*Id*" _
                And Not pf.Name Like "PI*Id*" Then
                pf.Orientation = Hidden
            End If
        End If
    Next
    'Turn on events
    Application.EnableEvents = True
End Sub

```

```

Sub m_RemoveAllEacFromValues()
    'Name of the source sheet
    Dim Sh_Pivot As String
    Dim WSh_Pivot As Worksheet
    Sh_Pivot = ActiveSheet.Name
    If Sh_Pivot <> "Pivot" Then
        MsgBox ("Select the sheet with the pivot table and restart the macro.")
        Exit Sub
    End If

```

```

Else
    Set WSh_Pivot = Sheets(Sh_Pivot)
End If
'pivot table
Dim pt As PivotTable
Dim Nof_Pt As Integer
Nof_Pt = 0
For Each pt In WSh_Pivot.PivotTables
    Nof_Pt = Nof_Pt + 1
Next pt
If Nof_Pt <> 1 Then
    MsgBox ("Make sure you habe only one Pivot in the sheet.")
    Exit Sub
Else
    Set pt = WSh_Pivot.PivotTables(1)
End If
'Turn off events
Application.EnableEvents = False
'data fields
Dim pf As PivotField
Dim Pos As Long
Pos = 1
On Error Resume Next
For Each pf In pt.PivotFields
    If pf.Name Like "*Eac*" Or pf.Name Like "*EAC*" Then
        pf.Orientation = xlHidden
        x = "Sum of " & pf.Name
        pt.PivotFields(x). _
            Orientation = xlHidden
    End If
Next
On Error GoTo 0
'Turn on events
Application.EnableEvents = True
End Sub

Sub test342()
    'Call z_CopySheet("1st step Planning Template", "Test")
    Sheets("1st step Planning Template").Activate
    Call z_CopyRange("All", "1st step Planning Template", _
        Sheets("1st step Planning Template").Range(Cells(1, 1), Cells(20, 38)), _
        "Test2", Sheets("test2").Range("A1"))
End Sub

Private Function z_CopyRange(PasteAllOrValues As String, Sh_from As String, Range_From As Range,
    _
        Sh_to As String, Cell_to As Range)
    'only works out if you invoke this function after this line!:Sheets(Sh_from).Activate
    'otherwise VBA cannot set the Range_From
    'set Cell_to as follows: Cell_to=Sheets(Sh_to).Range("A1")
    Sheets(Sh_from).Activate
    Range_From.Select

```

```

Selection.Copy
Sheets(Sh_to).Activate
Cell_to.Select
If PasteAllOrValues = "All" Then
    first = Range_From.Columns.End(xlToLeft).Column
    last = Range_From.Columns.End(xlToRight).Column
    Call z_Copy_ColWidth(Sh_from, Sh_to, first, last)
    Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
ElseIf PasteAllOrValues = "Values" Then
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
End If

End Function

Private Function z_Copy_ColWidth(Sh_from As String, Sh_to As String, Col_From, Col_to)
    For Col = Col_From To Col_to
        Sheets(Sh_to).Columns(Col).ColumnWidth = Sheets(Sh_from).Columns(Col).ColumnWidth
    Next Col
End Function

Private Sub z_CopySheet(Sh_from As String, Sh_to As String)
    Sheets(Sh_from).Select
    Cells.Select
    Application.CutCopyMode = False
    Selection.Copy
    Sheets(Sh_to).Select
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _
        False, Transpose:=False
    Range("A1").Select
    Sheets(Sh_from).Select
    Range("A1").Select
End Sub

Sub EventsOn()
    Application.EnableEvents = True
End Sub
Sub EventsOff()
    Application.EnableEvents = False
End Sub

Private Sub test()
    Dim Rng As Range
    Set Rng = Range(Cells(1, 48), Cells(130015, 48))
    'Set Rng = Range(Cells(10185, 48), Cells(10188, 48))
    Call z_DateCorrection(Rng)
End Sub

```



```

Function z_DateCorrection(Rng As Range)
    'select the range
    Rng.Select
    'define the replacements
    Dim aMonth As Variant
    aMonth = Array("JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV",
"DEC")
    'loop through all months
    Dim iMonth As Integer
    For iMonth = 1 To 12
        'Replacement rule
        Dim sMonth As String
        sMonth = aMonth(iMonth - 1)
        'perform the replacement
        Selection.Replace What:="-" & sMonth & "-", Replacement:="/" & CStr(iMonth) & "/", _
            LookAt:=xlPart, SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
            ReplaceFormat:=False
        'Simulate a double click into all cells to get date objects
        Selection.FormulaR1C1 = Selection.Value
    Next iMonth
End Function

```

```

Sub DataQualityMapping()
    Dim Sh_DataSet As String
    Dim Rng_DataSet_i As Range
    Dim Sh_MappingTable As String
    Dim Rng_MappingTable_i As Range
    Dim Col_Rng_DataSet_i As Long
    Dim RowSize_Rng_DataSet_i As Long
    Dim Col_Rng_MappingTable_i As Long
    Dim RowSize_Rng_MappingTable_i As Long
    '*****

    Sh_DataSet = "DataSet"
    Sh_MappingTable = "MappingTable"
    Col_Rng_DataSet_i = 4
    RowSize_Rng_DataSet_i = z_RowSize(Col_Rng_DataSet_i, Sh_DataSet)
    Col_Rng_MappingTable_i = 5
    RowSize_Rng_MappingTable_i = z_RowSize(Col_Rng_MappingTable_i, Sh_MappingTable)
    '*****

    Sheets(Sh_DataSet).Activate
    Set Rng_DataSet_i = Sheets(Sh_DataSet).Range(Cells(2, Col_Rng_DataSet_i), _
        Cells(RowSize_Rng_DataSet_i, Col_Rng_DataSet_i))
    Sheets(Sh_MappingTable).Activate
    Set Rng_MappingTable_i = Sheets(Sh_MappingTable).Range(Cells(2, Col_Rng_MappingTable_i), _
        Cells(RowSize_Rng_MappingTable_i, Col_Rng_MappingTable_i))

    Call z_DataQualityMapping(Sh_DataSet, Rng_DataSet_i, Sh_MappingTable, Rng_MappingTable_i)

End Sub

```

```

Function z_DataQualityMapping(Sh_DataSet As String, Rng_DataSet As Range, _

```

```

        Sh_MappingTable As String, Rng_MappingTable As Range)

Dim Adr As String
Address_Rng_MappingTable = Rng_MappingTable.Address
'Iterate through the rows
For Each Cell_DataSet In Rng_DataSet
    Rw = Cell_DataSet.Row
    V1 = Cell_DataSet.Value
    'if Cell_DataSet is found in Sh_MappingTable then perform the mapping
    If Not Sheets(Sh_MappingTable).Range(Address_Rng_MappingTable).Find(What:=Cell_DataSet,
LookAt:=xlWhole) Is Nothing Then
        'map
        Cell_DataSet.Offset(0, 0).Value = _
            Sheets(Sh_MappingTable).Range(Address_Rng_MappingTable).Find(What:=Cell_DataSet, _
LookAt:=xlWhole).Offset(0, 1)
    'if Cell_DataSet is not found in Sh_MappingTable
    Else
        'nothing to map
    End If
Next

End Function

```

```

Sub Addressbook1()
    'Reference Microsoft Outlook 14.0 Object Library

    Dim oAdrLst As Outlook.AddressList
    Dim oAdrEnt As Outlook.AddressEntries
    Dim objApp As New Outlook.Application

    Set objApp = CreateObject("Outlook.Application")

    'Return the personal address book.
    '1 contacts
    '2 suggested contacts
    '3 global address list
    Set oAdrLst = objApp.GetNamespace("MAPI").AddressLists(3)
    Set oAdrEnt = oAdrLst.AddressEntries

    Dim i As Long
    Dim strAddress() As String

    ReDim strAddress(0 To 4, 0 To oAdrEnt.Count - 1) As String

    For i = 1 To oAdrEnt.Count
        On Error Resume Next
        strAddress(0, i - 1) = oAdrEnt.Item(i).Name
        strAddress(1, i - 1) = oAdrEnt.Item(i).GetExchangeUser.Alias
        strAddress(2, i - 1) = oAdrEnt.Item(i).GetExchangeUser.JobTitle
        strAddress(3, i - 1) = oAdrEnt.Item(i).GetExchangeUser.CompanyName
        strAddress(4, i - 1) = oAdrEnt.Item(i).GetExchangeUser.PrimarySmtpAddress
        On Error GoTo 0
    Next i

```

```

Next
'Stop

Set fs = CreateObject("Scripting.FileSystemObject")
Set txt_file = fs.CreateTextFile("C:\Users\t740698\Desktop\testfile.txt", True)
For i = 1 To oAdrEnt.Count
    txt_file.WriteLine (strAddress(0, i - 1) & Chr(9) & strAddress(1, i - 1) & Chr(9) & strAddress(2, i - 1)
& Chr(9) & strAddress(3, i - 1) & Chr(9) & strAddress(4, i - 1))
Next i
'Stop

txt_file.Close

End Sub
Sub GetCurrentUser()
    Dim a(1 To 10) As String
    a(1) = (Outlook.Application.Session.CurrentUser.Name): Debug.Print a(1)
    a(2) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.Name):
Debug.Print a(2)
    a(3) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.Alias):
Debug.Print a(3)
    a(4) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.City): Debug.Print
a(4)
    a(5) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.Department):
Debug.Print a(5)
    a(6) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.JobTitle):
Debug.Print a(6)
    a(7) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.OfficeLocation):
Debug.Print a(7)
    a(8) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.StreetAddress):
Debug.Print a(8)
    a(9) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.CompanyName):
Debug.Print a(9)
    a(10) =
(Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.PrimarySmtpAddress):
Debug.Print a(10)

Set fs = CreateObject("Scripting.FileSystemObject")
Set txt_file = fs.CreateTextFile("C:\Users\t740698\Desktop\testfile.txt", True)
For i = 1 To 10
    txt_file.WriteLine (a(i))
Next i
txt_file.Close

End Sub

Sub Addressbook()

    Dim oAdrLst As Outlook.AddressList
    Dim oAdrEnt As Outlook.AddressEntries
    Dim objApp As New Outlook.Application

```

```

'Return the personal address book.
Set oAdrLst = objApp.GetNamespace("MAPI").AddressLists(1)
Set oAdrEnt = oAdrLst.AddressEntries

Dim i As Integer
Dim strAddress() As String

ReDim strAddress(oAdrEnt.Count - 1) As String

For i = 1 To oAdrEnt.Count
    strAddress(i - 1) = oAdrEnt.Item(i).Name
Next
Stop

Set fs = CreateObject("Scripting.FileSystemObject")
Set txt_file = fs.CreateTextFile("C:\Users\t740698\Desktop\testfile.txt", True)
For i = 1 To oAdrEnt.Count
    txt_file.WriteLine (strAddress(i - 1))
Next i
Stop

txt_file.Close
End Sub

Sub taskcustomer()
    Sheets("SmC_MasterDataSet_ResourceLevel").Activate
    tkcust_col = z_GetColumnIndex("TK Customer", 1, "SmC_MasterDataSet_ResourceLevel")
    PL1_Col = z_GetColumnIndex("Portfolio Level 1", 1, "SmC_MasterDataSet_ResourceLevel")
    PL2_Col = z_GetColumnIndex("Portfolio Level 2", 1, "SmC_MasterDataSet_ResourceLevel")
    For Each Cell In Range(Cells(2, tkcust_col), Cells(133094, tkcust_col))
        If Cells(Cell.Row, PL1_Col) = "SEEDS" Then
            If Cell.Value = Empty Then
                'Cell.Select
                If Cells(Cell.Row, PL2_Col) = "CORN_RD" Then
                    Cell.Value = "SE - CORN"
                End If
                If Cells(Cell.Row, PL2_Col) = "DFC_RD" Then
                    Cell.Value = "SE - DFC"
                End If
                If Cells(Cell.Row, PL2_Col) = "VEGETABLES_RD" Then
                    Cell.Value = "SE - VEG"
                End If
                If Cells(Cell.Row, PL2_Col) = "RICE_RD" Then
                    Cell.Value = "SE - RICE"
                End If
                If Cells(Cell.Row, PL2_Col) = "SOYBEAN_RD" Then
                    Cell.Value = "SE - SOY"
                End If
            End If
        End If
    Next

```

End Sub

Private flag As String

Private i As Integer

Sub m\_p()

    i = 1

    Call z\_p

End Sub

Function z\_p()

    Static iter As Integer

    iter = iter + i

    Stop

    If iter = 1 Then

        flag = "Goto-1"

    End If

    If flag = "Goto-1" Then

        a = 1

        Debug.Print "a=" & a

        flag = "Goto-2"

    End If

    If flag = "Goto-2" Then

        \*\*\*\*\*

        flag = "Goto-3"

        iter = iter + 1

        Debug.Print "iter" & iter

        Stop

        Call z\_SetOnTime("00:00:05", "z\_p")

        Exit Function

        \*\*\*\*\*

    End If

    If flag = "Goto-3" Then

        b = 1

        Debug.Print "b=" & b

        flag = "Goto-4"

    End If

    If flag = "Goto-4" Then

        c = 1

        Debug.Print "c=" & c

    End If

    iter = 0

End Function

Function z\_SetOnTime(Wait\_Sec As Variant, SubName As String)

    'input:Call z\_SetOnTime("00:00:05", "z\_SearchWindow")

    'call SearchWindow with ontime

    Dim TimerSet\_i As Variant

    TimerSet\_i = Now() + TimeValue(Wait\_Sec)

    Application.OnTime EarliestTime:=TimerSet\_i, Procedure:=SubName, Schedule:=True

    'now VBA is not running and waiting for the download window to pop up

End Function

Public Const CSIDL\_DESKTOP = &H0 ' Desktop (namespace root)  
Public Const CSIDL\_INTERNET = &H1 ' Internet virtual folder  
Public Const CSIDL\_PROGRAMS = &H2 ' Programs folder (under Start menu in [user] profile)  
Public Const CSIDL\_CONTROLS = &H3 ' Control Panel virtual folder  
Public Const CSIDL\_PRINTERS = &H4 ' Printers virtual folder  
Public Const CSIDL\_PERSONAL = &H5 ' Personal folder ([user] profile)  
Public Const CSIDL\_FAVORITES = &H6 ' Favorites folder ([user] profile)  
Public Const CSIDL\_STARTUP = &H7 ' Startup folder ([user] profile)  
Public Const CSIDL\_RECENT = &H8 ' Recent Documents folder ([user] profile)  
Public Const CSIDL\_SENDTO = &H9 ' SendTo folder ([user] profile)  
Public Const CSIDL\_DESKTOPDIRECTORY = &H10 ' Desktop folder ([user] profile)  
Public Const CSIDL\_DRIVES = &H11 ' My Computer virtual folder  
Public Const CSIDL\_NETWORK = &H12 ' Network Neighborhood root  
Public Const CSIDL\_NETHOOD = &H13 ' Network Neighborhood directory  
Public Const CSIDL\_FONTS = &H14 ' Fonts virtual folder  
Public Const CSIDL\_TEMPLATES = &H15 ' Templates folder ([user] profile)  
Public Const CSIDL\_COMMON\_STARTMENU = &H16 ' Start menu (All Users profile)  
Public Const CSIDL\_COMMON\_PROGRAMS = &H17 ' Programs folder (under Start menu in All Users profile)  
Public Const CSIDL\_COMMON\_STARTUP = &H18 ' Startup folder (All Users profile)  
Public Const CSIDL\_COMMON\_DESKTOPDIRECTORY = &H19 ' Desktop folder (All Users profile)  
Public Const CSIDL\_INTERNET\_CACHE = &H20 ' Internet Cache folder (Explorer 4.01 and Windows® 98).  
Public Const CSIDL\_COOKIES = &H21 ' Cookies folder  
Public Const CSIDL\_HISTORY = &H22 ' History folder  
Public Const CSIDL\_BITBUCKET = &HA ' Recycle Bin folder  
Public Const CSIDL\_STARTMENU = &HB ' Start menu ([user] profile)  
Public Const CSIDL\_APPDATA = &H1A ' Application Data ([user] profile) (Internet Explorer 4.0).  
Public Const CSIDL\_ALTSTARTUP = &H1D ' Alternate Startup ([user], DBCS)  
Public Const CSIDL\_COMMON\_ALTSTARTUP = &H1E ' Alternate Startup folder (All Users profile, DBCS)  
Public Const CSIDL\_COMMON\_FAVORITES = &H1F ' Favorites folder (All Users profile)  
Public Const CSIDL\_PRINTHOOD = &H1B ' PrintHood folder ([user] profile)  
Public Const CSIDL\_MYPICTURES = &H27 ' My Pictures folder (Windows 2000 & Windows Me).  
Public Const CSIDL\_COMMON\_ADMINTOOLS = &H2F ' Administrative tools (All Users profile) (Windows 2000 & Windows Me).  
Public Const CSIDL\_COMMON\_DOCUMENTS = &H2E ' Documents folder (All Users profile)  
Public Const CSIDL\_ADMINTOOLS = &H30 ' Administrative Tools ([user] profile) (Windows 2000 & Windows Me).  
Public Const CSIDL\_PROGRAM\_FILES = &H26 ' Program Files folder (Windows 2000 & Windows Me).  
Public Const CSIDL\_PROGRAM\_FILES\_COMMON = &H2B ' Common Files folder (Windows 2000 & Windows Me).  
Public Const CSIDL\_COMMON\_APPDATA = &H23 ' Application data for all users. A typical path is C:\Documents and Settings\All Users\Application (Windows 2000 & Windows Me)  
Public Const CSIDL\_COMMON\_TEMPLATES = &H2D ' File system directory that contains the templates that are available to all users. A typical path is C:\Documents and Settings\All Users\Templates. Valid only for Windows NT systems.  
Public Const CSIDL\_CONNECTIONS = &H31 ' Virtual folder containing Network and Dial-up connections

Public Const CSIDL\_LOCAL\_APPDATA = &H1C ' File system directory that serves as a data repository for local (nonroaming) applications. A typical path is C:\Documents and Settings\username\Local Settings\Application Data (Windows 2000 & Windows Me).

Public Const CSIDL\_PROFILE = &H28 'User's profile folder (Windows 2000 & Windows Me).

Public Const CSIDL\_PROGRAM\_FILES\_COMMONX86 = &H2C ' The x86 Program Files Common folder on RISC systems.

Public Const CSIDL\_PROGRAM\_FILESX86 = &H2A ' The x86 Program Files folder on RISC systems.

Public Const CSIDL\_SYSTEM = &H25 ' System folder. A typical path is C:\WINNT\SYSTEM32 (Windows 2000 & Windows Me).

Public Const CSIDL\_SYSTEMX86 = &H29 ' The x86 system directory on RISC systems.

Public Const CSIDL\_WINDOWS = &H24 ' Windows directory or SYSROOT. This corresponds to the %windir% or %SYSTEMROOT% environment variables. A typical path is C:\WINNT (Windows 2000 & Windows Me).

Public Const CSIDL\_MYMUSIC = &HD ' File system directory that serves as a common repository for music files.

Public Const CSIDL\_MYVIDEO = &HE ' File system directory that serves as a common repository for video files.

Public Const CSIDL\_COMMON\_MUSIC = &H35 ' My Music folder for all users. See CSIDL\_MYMUSIC.

Public Const CSIDL\_COMMON\_PICTURES = &H36 ' My Pictures folder for all users. See CSIDL\_MYPICURES.

Public Const CSIDL\_COMMON\_VIDEO = &H37 ' My Video folder for all users. See CSIDL\_MYVIDEO.

Public Const CSIDL\_RESOURCES = &H38 ' System resource directory. A typical path is C:\WINNT\Resources.

Public Const CSIDL\_RESOURCES\_LOCALIZED = &H39 ' Localized resource directory. See CSIDL\_RESOURCES.

Public Const CSIDL\_COMMON\_OEM\_LINKS = &H3A ' Folder containing links to All Users OEM specific applications.

Public Const CSIDL\_CDBURN\_AREA = &H3B ' File system folder used to hold data for burning to a CD. Typically [User Profile Folder]\Local Settings\Applications Data\Microsoft\CD Burning.

Public Const CSIDL\_COMPUTERSNEARME = &H3D ' Computers Near Me folder. Virtual folder containing links to "nearby" computers on the network. Nearness it is established by common workgroup membership.

Public Const MAX\_PATH = 260

Public Const NOERROR = 0

Public Type shiEMID

cb As Long

abID As Byte

End Type

Public Type ITEMIDLIST

mkid As shiEMID

End Type

Declare Function SHGetSpecialFolderLocation Lib "shell32.dll" (ByVal hwndOwner As Long, ByVal nFolder As Long, pidl As ITEMIDLIST) As Long

Declare Function SHGetPathFromIDList Lib "shell32.dll" Alias "SHGetPathFromIDListA" (ByVal pidl As Long, ByVal pszPath As String) As Long

Declare Function getUserNam Lib "advapi32.dll" Alias "GetUserNameA" (ByVal lpBuffer As String, nSize As Long) As Long

Sub tests()

desk = GetSpecialfolder(CSIDL\_DESKTOP)

'or

desk = GetSpecialfolder(CSIDL\_DESKTOPDIRECTORY)

```

'or
desk = GetSpecialFolder(CSIDL_COMMON_DESKTOPDIRECTORY)
End Sub

Public Function GetSpecialFolder(CSIDL As Long) As String
    Dim IDL As ITEMIDLIST
    Dim sPath As String
    Dim iReturn As Long

    iReturn = SHGetSpecialFolderLocation(100, CSIDL, IDL)

    If iReturn = NOERROR Then
        sPath = Space(512)
        iReturn = SHGetPathFromIDList(ByVal IDL.mkid.cb, ByVal sPath)
        sPath = RTrim$(sPath)
        If Asc(Right(sPath, 1)) = 0 Then sPath = Left$(sPath, Len(sPath) - 1)
        GetSpecialFolder = sPath
        Exit Function
    End If
    GetSpecialFolder = ""
End Function

Sub test()
    a = getDesktopFolder
    MsgBox a
    b = getUserName1
    MsgBox b
    c = getUserName2
    MsgBox c
End Sub

Function getDesktopFolder() As String
    Dim s As String
    s = CreateObject("WScript.Shell").SpecialFolders("Desktop")
    getDesktopFolder = s
End Function

Function getUserName2() As String
    Dim s As String
    Dim x As Variant
    Dim User As String
    s = CreateObject("WScript.Shell").SpecialFolders("Desktop")
    x = Split(s, "\")
    User = x(UBound(x) - 1)
    getUserName2 = User
End Function

Function getUserName1() As String
    Dim Puffer As String * 256
    Dim User As String
    Dim Ret As Long
    Ret = getUserName(Puffer, Len(Puffer))

```



```

If Ret <> 0 Then
    User = left$(Puffer, InStr(1, Puffer, vbNullChar) - 1)
    getUsername1 = User
End If
End Function

```

```

Sub ShowFileList(folderspec)
    Dim fs, f, f1, fc, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(folderspec)

    Set fc = f.Files
    For Each f1 In fc
        s = s & f1.Name
        s = s & vbCrLf
    Next
    MsgBox s
End Sub

```

```

Sub ShowDriveList()
    Dim fs, d, dc, s, n
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set dc = fs.Drives
    For Each d In dc
        s = s & d.DriveLetter & " - "
        If d.DriveType = 3 Then
            n = d.ShareName
        Else
            n = d.VolumeName
        End If
        s = s & n & vbCrLf
    Next
    MsgBox s
End Sub

```

```

Sub Macro1()
    '
    pathspec = "C:"

```

```

Call AbsolutePath(pathspec)
'
End Sub

```

```

Sub AbsolutePath(pathspec)
    Dim fs, f, f1, fc, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetAbsolutePathName("C:\Users\t740698\Desktop\Task_Jochen")

```

```

    MsgBox f
End Sub
Sub ShowFileAccessInfo(filespec)
    Dim fs, d, f, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(filespec)
    s = UCase(f.path) & vbCrLf
    s = s & "Created: " & f.DateCreated & vbCrLf
    s = s & "Last Accessed: " & f.DateLastAccessed & vbCrLf
    s = s & "Last Modified: " & f.DateLastModified
    MsgBox s, 0, "File Access Info"
End Sub

```

```

'Sub Macro1()
' Dim Sh_DataSetBackup As Worksheet
' Set Sh_DataSetBackup = Sheets("DataSetBackup")
' For i = 5 To 60
' Sh_DataSetBackup.Range("G" & CStr(i)).Interior.ColorIndex = i
' Next
'End Sub
'
'Sub GetCurrentUser()
' Dim a() As String
' a(1) = (Outlook.Application.Session.CurrentUser.Name): Debug.Print a(1)
' a(2) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.Name):
Debug.Print a(2)
' a(3) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.Alias):
Debug.Print a(3)
' a(4) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.City): Debug.Print
a(4)
' a(5) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.Department):
Debug.Print a(5)
' a(6) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.JobTitle):
Debug.Print a(6)
' a(7) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.OfficeLocation):
Debug.Print a(7)
' a(8) = (Outlook.Application.Session.CurrentUser.AddressEntry.GetExchangeUser.StreetAddress):
Debug.Print a(8)
'
' Set fs = CreateObject("Scripting.FileSystemObject")
' Set txt_file = fs.CreateTextFile("C:\Users\t740698\Desktop\testfile.txt", True)
' For i = 1 To 9
' txt_file.WriteLine (a(i))
' Next i
' txt_file.Close
'End Sub
'
'Sub EnumerateFoldersInStores()
' Dim olApp As New Outlook.Application
' Dim colStores As Outlook.Stores

```

```

' Dim oStore As Outlook.Store
' Dim oRoot As Outlook.Folder
'
' On Error Resume Next
' Set colStores = olApp.Session.Stores
' For Each oStore In colStores
'     Set oRoot = oStore.GetRootFolder
'     Debug.Print (oRoot.FolderPath)
'     EnumerateFolders oRoot
' Next
'End Sub

```

```

'Private Sub EnumerateFolders(ByVal oFolder As Outlook.Folder)
' Dim folders As Outlook.folders
' Dim Folder As Outlook.Folder
' Dim foldercount As Integer
'
' On Error Resume Next
' Set folders = oFolder.folders
' foldercount = folders.Count
' 'Check if there are any folders below oFolder
' If foldercount Then
'     For Each Folder In folders
'         Debug.Print (Folder.FolderPath)
'         Debug.Print (Folder.Items.Count)
'         EnumerateFolders Folder
'     Next
' End If
'End Sub

```

```

Sub a1()
    Dim txt_file As Object
    Call TextFile_Create(txt_file, "C:\Users\t740698\Desktop\testfile.txt")
    Call TextFile_Open_Or_CreateAndOpen(txt_file, "C:\Users\t740698\Desktop\testfile.txt")
    Dim txt(0) As Variant
    txt(0) = "Start - Crop Split: " & CStr(Now())
    Call TextFile_WriteInto1(txt_file, txt)
    Call TextFile_Close(txt_file)
End Sub

```

```

Sub a2()
    Dim txt_file As Object
    Call TextFile_Create(txt_file, "C:\Users\t740698\Desktop\testfile.txt")
    Call TextFile_Close(txt_file)
    Call TextFile_Open_Or_CreateAndOpen(txt_file, "C:\Users\t740698\Desktop\testfile.txt")
    Call TextFile_WriteInto2(txt_file, Array("txt", "test"))
    Call TextFile_Close(txt_file)
End Sub

```

```

Function TextFile_Create(ByRef txt_file As Object, PathAndName As String)
    'call
    'Call TextFile_Create(txt_file, "C:\Users\t740698\Desktop\testfile.txt")
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set txt_file = fs.CreateTextFile(PathAndName, True)
End Function

```

```

Function TextFile_Open_Or_CreateAndOpen(ByRef txt_file As Object, PathAndName As String)
'Call:
'Call TextFile_Open_Or_CreateAndOpen(txt_file, "C:\Users\t740698\Desktop\testfile.txt")
'if an error message pops up then the file is already open
Set fs = CreateObject("Scripting.FileSystemObject")
'8=For Appending at the end of the file
'True=Create a new file if it does not exist
Set txt_file = fs.OpenTextFile(PathAndName, 8, True)
End Function
Function TextFile_WriteInto1(txt_file As Object, txt_arr() As Variant)
'Call:
'Dim txt(0 to 2) As Variant
'txt(0) = "Start - Task1: " & CStr(Now())
'txt(1) = "Stop - Task1: " & CStr(Now())
'txt(2) = "Start - Task2: " & CStr(Now())
'Call TextFile_WriteInto1(txt_file, txt)
For i = LBound(txt_arr) To UBound(txt_arr)
    txt_file.WriteLine (txt_arr(i))
Next i
End Function
Function TextFile_WriteInto2(txt_file As Object, txt_arr As Variant)
'Call:
'Call TextFile_WriteInto2(txt_file, Array("string1 " & CStr(Now()), "string2 " & CStr(Now())))
For i = LBound(txt_arr) To UBound(txt_arr)
    txt_file.WriteLine (txt_arr(i))
Next i
End Function
Function TextFile_Close(txt_file As Object)
    txt_file.Close
End Function

```

## **File: PivotsMacros**

```

Sub SmC_TimeCardActuals()
'Macro generated by Roland.Benz@Syngenta.com and franz.schuermann@syngenta.com (PMEC,
Project Management Excellence)
'Date: 26.9.2011
'Macro for Lee Hubbard
Dim start As Date: Dim Duration As Date
start = Now()

'MsgBox to activate the right sheet, &vbCrLf &
If 1 Then
    Response = MsgBox( _
        "Please select the Excel sheet you want the macro to be applied to!" & Chr(13) & _
        "Click Yes if the right sheet is already selected." & Chr(13) & _
        "Otherwise click No, select the Excel sheet by clicking on its tab" & Chr(13) & _
        "                                at the botton and then restart the macro.", _
        vbYesNo)
End If

```

'Execute the tasks

'Copy the input file ActiveSheet.Name in a new file Sh\_Source and make changes

'Make new sheets Sh\_Pivot and make the pivots

If Response = vbYes Then

'Make a copy of the input sheet, change the sheet tab color and then make some some changes

Dim Sh As String

Sh = ActiveSheet.Name ' msgbox asks to make the right sheet active

Dim Sh\_Source As String

Sh\_Source = "ActualsByWeek"

If 1 Then

Call z\_ShNewFlatValueCopy(Sh, Sh\_Source, "End")

'change the colour of the sheet name on the tab

With ActiveWorkbook.Sheets(Sh\_Source).Tab

.Color = RGB(0, 32, 90)

.TintAndShade = 0

End With

End If

If 1 Then

'Add a column for person site information

Sheets(Sh\_Source).Select

Rows("1:1").Find(What:="EMPLOYEE\_NAME", LookAt:=xlWhole).Select

ActiveCell.Offset(0, 1).EntireColumn.Insert

ActiveCell(1, 2).Value = "SITE"

'Fill the new column with person site info

Range(ActiveCell(2, 1), ActiveCell(2, 1).End(xlDown)).Select

For Each rSite In Selection.Cells

rSite.Offset(0, 1).Value = Right(rSite.Value, 4)

Next

End If

If 1 Then

'Add a column for a concatenated string of PI ID and Task ID

Sheets(Sh\_Source).Select

Rows("1:1").Find(What:="STAFF\_DAYS", LookAt:=xlWhole).Select

ActiveCell.Offset(0, 1).EntireColumn.Insert

ActiveCell(1, 2).Value = "IDENTIFICATION"

Range(ActiveCell(2, 2), ActiveCell(2, 1).End(xlDown).Offset(0, 1)).Select

For Each rSite In Selection.Cells

rSite.Value = rSite.Offset(0, -16).Value & "-" & rSite.Offset(0, -12).Value

Next

Range("A1").Select

End If

'Make new sheets and the pivots from Sh\_Source

Dim Sh\_Pivot As String

Dim RowU, Coll, RowD, ColR As Long

Dim Piv\_RowD As Long

Dim Source\_Rng As Range

Dim Piv\_sULCell As String

Dim Piv\_ULCell As Range

Dim Piv\_F\_R\_C\_V(0 To 3) As Variant

Dim PivName(0 To 2) As String

```

'Make a new sheet and add a pivot
'Name of the sheet
Sh_Pivot = "TimeToDateByTask"
If 1 Then
    Call z_ShNew(Sh_Pivot, "End")
    'change the colour of the sheet name on the tab
    With ActiveWorkbook.Sheets(Sh_Pivot).Tab
        .Color = RGB(255, 255, 0)
        .TintAndShade = 0
    End With
End If
If 1 Then
    'Parameters for the z_GeneratePivotTable function and its call
    'Name of the pivot sheet
    'Sh_Pivot = "Time-to-Date-By-Task-RB"
    'Name of the source sheet
    'Sh_Source = "ActualsByWeek-RB"
    'Determine the range of Sh_Source and create a range object
    RowU = 1: ColL = 1
    RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
    Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
    'Create a range object for the upper left corner of the pivot
    Piv_sULCell = "B9"
    Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
    'Determine the pivot fields
    Piv_F_R_C_V(0) = Array(13, 9, 11, 7, 12, 10) 'filter
    Piv_F_R_C_V(1) = Array(17, 5) 'row labels
    Piv_F_R_C_V(2) = Array() 'column labels
    Piv_F_R_C_V(3) = Array(16) 'values
    'Call the function that creates the pivot
    PivName(0) = z_GeneratePivotTable(Piv_ULCell, Source_Rng, Sh_Source, Sh_Pivot, Piv_F_R_C_V)
End If
If 1 Then
    'Pivot PivName(0)
    'set the filters
    ActiveSheet.PivotTables(PivName(0)).PivotFields("RESOURCE_YEAR").ClearAllFilters
    ActiveSheet.PivotTables(PivName(0)).PivotFields("RESOURCE_YEAR").CurrentPage = "2011"
    'change the column width of col A
    Columns("A:A").ColumnWidth = 5
    'insert formula into the Sh_Pivot
    Sheets(Sh_Pivot).Select
    Range("D7").Select
    Selection.Formula = "=GETPIVOTDATA("""STAFF_DAYS""",$B$9)"
    'format columns in Sh_Pivot
    Range("D2:D7").Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent3
        .TintAndShade = 0.599993896298105
        .PatternTintAndShade = 0
    End With

```

```

Columns("A:A").ColumnWidth = 3
Columns("D:D").Select
Selection.NumberFormat = "#,##0.0"
ActiveWorkbook.Save
Range("B9").Select
End If

```

'Make a new sheet and add two pivots

'Name of the sheet

Sh\_Pivot = "DataQualityLocation"

If 1 Then

Call z\_ShNew(Sh\_Pivot, "End")

'change the colour of the sheet name on the tab

With ActiveWorkbook.Sheets(Sh\_Pivot).Tab

.Color = RGB(243, 130, 37)

.TintAndShade = 0

End With

End If

If 1 Then

'Parameters for the z\_GeneratePivotTable function and its call

'Name of the pivot sheet

'Sh\_Pivot = "DataQualityLocation-RB"

'Name of the source sheet

'Sh\_Source = "ActualsByWeek-RB"

'Determine the range of Sh\_Source and create a range object

RowU = 1: ColL = 1

RowD = z\_RowSize(1, Sh\_Source): ColR = z\_ColSize(1, Sh\_Source)

Set Source\_Rng = Sheets(Sh\_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))

'Create a range object for the upper left corner of the pivot

Piv\_sULCell = "B7"

Set Piv\_ULCell = Sheets(Sh\_Pivot).Range(Piv\_sULCell)

'Determine the pivot fields

Piv\_F\_R\_C\_V(0) = Array(13, 9) 'filter (enter numbers in proper order)

Piv\_F\_R\_C\_V(1) = Array(12, 19) 'row labels (enter numbers in reverse order)

Piv\_F\_R\_C\_V(2) = Array(7) 'column labels

Piv\_F\_R\_C\_V(3) = Array(16) 'values

'Call the function that creates the pivot

PivName(1) = z\_GeneratePivotTable(Piv\_ULCell, Source\_Rng, Sh\_Source, Sh\_Pivot, Piv\_F\_R\_C\_V)

End If

If 1 Then

'Parameters for the z\_GeneratePivotTable function and its call

'Name of the sheet

'Sh\_Pivot = "DataQualityLocation-RB"

'Determine the range of Sh\_Source and create a range object

RowU = 1: ColL = 1

RowD = z\_RowSize(1, Sh\_Source): ColR = z\_ColSize(1, Sh\_Source)

Set Source\_Rng = Sheets(Sh\_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))

'Create a range object for the upper left corner of the pivot

Piv\_RowD = z\_RowSize(2, Sh\_Pivot)

Piv\_sULCell = "B" & Piv\_RowD + 8 'place the second pivot under the first pivot

Set Piv\_ULCell = Sheets(Sh\_Pivot).Range(Piv\_sULCell)

'Determine the pivot fields

Piv\_F\_R\_C\_V(0) = Array(13, 9) 'filter (enter numbers in proper order)

```

Piv_F_R_C_V(1) = Array(17, 11, 19) 'row labels (enter numbers in reverse order)
Piv_F_R_C_V(2) = Array(7) 'column labels
Piv_F_R_C_V(3) = Array(16) 'values
'Call the function that creates the pivot
PivName(2) = z_GeneratePivotTable(Piv_ULCell, Source_Rng, Sh_Source, Sh_Pivot, Piv_F_R_C_V)
End If
'Add some formats and texts
If 1 Then
    'Pivot PivName(1)
    'set the filters
    ActiveSheet.PivotTables(PivName(1)).PivotFields("RESOURCE_YEAR").ClearAllFilters
    ActiveSheet.PivotTables(PivName(1)).PivotFields("RESOURCE_YEAR").CurrentPage = "2011"
    ActiveSheet.PivotTables(PivName(1)).PivotFields("TASK_LOCATION").ClearAllFilters
    ActiveSheet.PivotTables(PivName(1)).PivotFields("TASK_LOCATION").CurrentPage = "(blank)"
    'change the column width of col A
    Columns("A:A").ColumnWidth = 5
    'add some text
    Range("B2").Select
    ActiveCell.Value = "Number of Tasks with recorded staff-days, Task Location =0"
    Range("E2").Select
    ActiveCell.Value = "241"
    'alignment
    Range("B2:D2").Select
    With Selection
        .HorizontalAlignment = xlLeft
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    'merge cells
    Range("B2:D2").Select
    Selection.Merge
    'change the font
    Range("B2:E2").Select
    Selection.Font.Bold = True
    'add borders
    Range("B2:E2").Select
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThick
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = 0

```



```

        .TintAndShade = 0
        .Weight = xlThick
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThick
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThick
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThick
    End With
    With Selection.Borders(xlInsideHorizontal)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThick
    End With
    'freeze panes
    Rows("9:9").Select
    ActiveWindow.FreezePanes = True
    'Pivot PivName(2)
    'copy/paste some text
    Range("B2:E2").Select
    Selection.Copy
    Range("B" & Piv_RowD + 3).Select
    ActiveSheet.Paste
    'add filters
    ActiveSheet.PivotTables(PivName(2)).PivotFields("RESOURCE_YEAR").ClearAllFilters
    ActiveSheet.PivotTables(PivName(2)).PivotFields("RESOURCE_YEAR").CurrentPage = "2011"
    ActiveSheet.PivotTables(PivName(2)).PivotFields("TASK_LOCATION").ClearAllFilters
    ActiveSheet.PivotTables(PivName(2)).PivotFields("TASK_LOCATION").CurrentPage = "(blank)"
    Range("B7").Select
End If
Else
    Exit Sub
End If

'save the workbook
'ActiveWorkbook.Save
Duration = Now() - start
Debug.Print Duration
End Sub

```

Sub z\_ShNewFlatValueCopy(Sh As String, Sh\_new As String, Optional Where As String, Optional ByRef Sh\_Ref As Worksheet)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Name of the Sh to copy, Name of the new Sh\_new, where to place the new Sh\_new, before or after some Sh\_Ref, at the begin or the end

Call z\_ShAdd(Where, Sh\_Ref)

On Error Resume Next

ActiveSheet.Name = Sh\_new

If Err.Number <> 0 Then

Application.DisplayAlerts = False

ActiveSheet.Delete

Application.DisplayAlerts = True

Sheets(Sh\_new).Cells.ClearContents

End If

On Error GoTo 0

Sheets(Sh).Cells.Copy

Sheets(Sh\_new).Range("A1").Select

Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks \_  
:=False, Transpose:=False

Sheets(Sh\_new).Columns("A:GA").ColumnWidth = 20

End Sub

Function z\_ShNew(Sh As String, Optional Where As String, Optional ByRef Sh\_Ref As Worksheet, Optional ByRef Wb As Workbook)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Name of the new Sh, where to place the new Sh, before or after some Sh\_Ref, at the begin or the end

On Error Resume Next

WorksheetExists = (Sheets(Sh).Name <> "")

On Error GoTo 0

If WorksheetExists = False Then

Call z\_ShAdd(Where, Sh\_Ref)

ActiveSheet.Name = Sh 'Worksheets.Add(Before:=Worksheets(1)).Name = Sh

End If

'Clear contents

Sheets(Sh).Activate

ActiveSheet.Cells.Select

Selection.ClearContents

'Format

Sheets(Sh).Columns.ColumnWidth = 20

Sheets(Sh).Rows.RowHeight = 15

End Function

Function z\_ShAdd(Optional Where As String, Optional ByRef Sh\_Ref As Worksheet)

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: where to place the added Sh, before or after some Sh\_Ref, at the begin or the end

If Not Sh\_Ref Is Nothing Then

If Where = ("Before:=") Then

Sheets.Add Sh\_Ref

```

Elseif Where = ("After:=") Then
    Sheets.Add , Sh_Ref
Else
    Sheets.Add Before:=Sheets(1)
End If
Else
    If Where = ("End") Then
        Sheets.Add After:=Sheets(Sheets.Count)
    Elseif Where = ("Begin") Then
        Sheets.Add Before:=Sheets(1)
    Else
        Sheets.Add Before:=Sheets(1)
    End If
End If
End Function

```

```

Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef MyWb As Workbook)
As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: column, Output: row with the last entry in that column
'SearchCol datatype changed from integer
'Activate the Sheet
Sheets(Sh).Activate
'Determine the row size
z_RowSize = If(IsEmpty(Cells(1048576, SearchCol)), Cells(1048576, SearchCol).End(xlUp).Row,
1048576)
End Function

```

```

Function z_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef MyWb As Workbook)
As Long
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Row, Output: column with the last entry in that row
'SearchCol datatype changed from integer
'Activate the Sheet
Sheets(Sh).Activate
'Determine the col size
z_ColSize = If(IsEmpty(Cells(SearchRow, 16384)), Cells(SearchRow, 16384).End(xlToLeft).Column,
16384)
End Function

```

```

Function z_GeneratePivotTable(Piv_ULCell As Range, Source_Rng As Range, _
    Sh_Source As String, Sh_Pivot As String, Piv_F_R_C_V As Variant) As String
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input parameters (arguments) for the z_GeneratePivotTable function and its call
'Name of the pivot sheet
'Sh_Pivot = "TimeToDateByTask"
'Name of the source sheet
'Sh_Source = "ActualsByWeek"

```

```

'Determine the range of Sh_Source and create a range object
'RowU = 1: ColL = 1
'RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
'Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'Create a range object for the upper left corner of the pivot
'Piv_sULCell = "B9"
'Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
'Determine the pivot fields
'Piv_F_R_C_V(0) = Array(13, 9, 11, 7, 12, 10) 'filter
'Piv_F_R_C_V(1) = Array(17, 5) 'row labels
'Piv_F_R_C_V(2) = Array() 'column labels
'Piv_F_R_C_V(3) = Array(16) 'values

'Creat the strings for the function ActiveWorkbook.PivotCaches.Create() further below
Dim sSource_Rng As String
sSource_Rng = Sh_Source & "!" & Source_Rng.Address(ReferenceStyle:=xlR1C1)
Dim sPivot_Rng As String
sPivot_Rng = Sh_Pivot & "!" & Piv_ULCell.Address(ReferenceStyle:=xlR1C1)

'Store the range.address information into an array
Dim RngAddress As Variant
RngAddress = z_RangeAddressAsArray(Source_Rng)

'Store the Column names into an array
ReDim PivChosenField(RngAddress(2) To RngAddress(4)) As String
For i = RngAddress(2) To RngAddress(4)
    PivChosenField(i) = Source_Rng.Cells(1, i)
Next i

'Store the column names of the ReportFilter, RowLabel, ColLabel and Value into arrays
ReDim PivReportFilter(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivRowLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivColLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivValue(RngAddress(2) - 1 To RngAddress(4) - 1) As String
For i = RngAddress(2) - 1 To RngAddress(4) - 1
    On Error Resume Next
    PivReportFilter(i) = PivChosenField(Piv_F_R_C_V(0)(i))
    On Error GoTo 0
    On Error Resume Next
    PivRowLabel(i) = PivChosenField(Piv_F_R_C_V(1)(i))
    On Error GoTo 0
    On Error Resume Next
    PivColLabel(i) = PivChosenField(Piv_F_R_C_V(2)(i))
    On Error GoTo 0
    On Error Resume Next
    PivValue(i) = PivChosenField(Piv_F_R_C_V(3)(i))
    On Error GoTo 0
Next i

'generate Pivot
Sheets(Sh_Pivot).Select
Piv_ULCell.Select
ActiveWorkbook.PivotCaches.Create( _

```

```
Source:=xlDatabase, _  
SourceData:=sSource_Rng, _  
Version:=xlPivotTableVersion12).CreatePivotTable _  
TableDestination:=sPivot_Rng, _  
TableName:=Piv_Name, _  
DefaultVersion:=xlPivotTableVersion12
```

'get Pivot table name

'if PivName = "PivotTable" is used more than once an iteger is added to the name

PivName = ActiveSheet.PivotTables(1).Name

For i = RngAddress(2) - 1 To RngAddress(4) - 1

'Define row labels

On Error Resume Next

With ActiveSheet.PivotTables(PivName).PivotFields(PivRowLabel(i))

.Orientation = xlRowField

.Position = 1

End With

On Error GoTo 0

'Define column labels

On Error Resume Next

With ActiveSheet.PivotTables(PivName).PivotFields(PivCollabel(i))

.Orientation = xlColumnField

.Position = 1

End With

On Error GoTo 0

'Define values

On Error Resume Next

ActiveSheet.PivotTables(PivName).AddDataField ActiveSheet.PivotTables( \_

PivName).PivotFields(PivValue(i)), "Sum of STAFF\_DAYS", xlSum

On Error GoTo 0

'Define report filters

On Error Resume Next

With ActiveSheet.PivotTables(PivName).PivotFields(PivReportFilter(i))

.Orientation = xlPageField

.Position = 1

End With

On Error GoTo 0

Next i

'Change the layout

With ActiveSheet.PivotTables(PivName)

.InGridDropZones = True

.RowAxisLayout xlTabularRow

End With

'Define all columns from : Choose fields to add to report

For i = RngAddress(2) To RngAddress(4)

ActiveSheet.PivotTables(PivName).PivotFields(PivChosenField(i)).Subtotals = \_

Array(False, False, False, False, False, False, False, False, False, False, False)

Next i

'output

z\_GeneratePivotTable = PivName

End Function

Function z\_RangeAddressAsArray(Rng As Range) As Variant

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Range(Cells(a,b),Cells(c,d)), Output: Array(a,b,c,d)

sRngAddress = Rng.Address(ReferenceStyle:=xlR1C1)

Dltrlst = Array("\$", "R", "C", ":")

Rpllst = Array("@", "@", "@", "@")

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

    If Rpllst(k) = Empty Then

        l = Rpllst(0)

    Else

        l = k

    End If

    sRngAddress = Replace(sRngAddress, Dltrlst(k), Rpllst(l))

Next k

sRngAddress = Replace(sRngAddress, Rpllst(0) & Rpllst(0), Rpllst(l))

Dim RngAddress As Variant

RngAddress = Split(sRngAddress, Rpllst(0))

For i = 1 To 4 Step 1

    RngAddress(i - 1) = RngAddress(i)

Next i

ReDim Preserve RngAddress(1 To 4)

z\_RangeAddressAsArray = RngAddress

End Function

## **File: TimeCard ActualsByWeek V2**

Sub SmC\_CleanRBS\_Extract()

'A user of this macro needs to provide an extract of the SmartChoice RBS which is limited to 4 levels.

'Otherwise the macro will stop and ask the user to first generate such an extract.

Dim IntRBSValue As String, IntRBSValue2 As String

'Column TimeCardManager must exist and must be blank for all rows. Level 1-4 is reserved for RBS

'Level 5-.. can be (mis-)used for other tasks

If Not WorksheetFunction.CountBlank(Range(Rows("1:1").Find(What:="TimeCard Manager",

LookAt:=xlWhole).Offset(1, 0), Rows("1:1").Find(What:="TimeCard Manager",

LookAt:=xlWhole).Offset(1000, 0))) = Range(Rows("1:1").Find(What:="TimeCard Manager",

LookAt:=xlWhole).Offset(1, 0), Rows("1:1").Find(What:="TimeCard Manager",

LookAt:=xlWhole).Offset(1000, 0)).Count Then

    MsgBox ("Please generate first a SmC RBS extract which is limited to level 4 before running this macro")

    Exit Sub

End If

```

'MsgBox to activate the right sheet, &vbCrLf &
If 1 Then
    Response = MsgBox( _
        "Please select the Excel sheet you want the macro to be applied to!" & Chr(13) & _
        " " & Chr(13) & _
        "Click Yes: if the right sheet is already selected" & Chr(13) & _
        "      Otherwise" & Chr(13) & _
        "Click No: and select the Excel sheet by clicking on its tab at the" & Chr(13) & _
        "      bottom and then restart the macro", _
        vbYesNo)
End If

If Response = vbYes Then

    'The active sheet will be renamed and a new sheet RBS Table will be created
    ActiveSheet.Name = "RBS Extract"
    Sheets("RBS Extract").Copy Before:=Sheets("RBS Extract")
    ActiveSheet.Name = "RBS Table"

    ' Rows("2:2").Select
    ' ActiveWindow.FreezePanes = True

    'Generally "unmerge" merged cells
    Cells.Select
    Selection.MergeCells = False

    'Remove the ALL level
    If Cells(2, 7).Value = "ALL" Then
        Cells(2, 7).EntireRow.Delete
    End If

    'Remove not used columns
    'Columns("A:B").EntireColumn.Delete
    'Columns("E:E").EntireColumn.Delete
    'Columns("A:A").EntireColumn.Delete

    'Add headers for the columns
    Range("A:H").ColumnWidth = 20
    Cells(1, 1).Value = "RBS Level 1 Name"
    Cells(1, 3).Value = "RBS Level 2 Name"
    Cells(1, 5).Value = "RBS Level 3 Name"
    Cells(1, 2).Value = "RBS Level 1 Description"
    Cells(1, 4).Value = "RBS Level 2 Description"
    Cells(1, 6).Value = "RBS Level 3 Description"

    'Assign RBS level 1, 2 and 3 values to the cells on the left of each entry
    'RBS level 1
    Cells(2, 5).Select
    Do While Not ActiveCell.Value = ""
        If Not ActiveCell(1, -2).Borders(xlEdgeRight).LineStyle = xlContinuous Then
            IntRBSValue = ActiveCell.Value
            IntRBSValue2 = ActiveCell(1, 2).Value
            ActiveCell.EntireRow.Delete
        End If
    Loop

```

```

        Do While ActiveCell(1, -2).Borders(xlEdgeRight).LineStyle = xlContinuous
            ActiveCell(1, -3).Value = IntRBSValue
            ActiveCell(1, -2).Value = IntRBSValue2
            ActiveCell(2, 1).Select
        Loop
    End If
Loop

'RBS level 2
Cells(2, 5).Select
Do While Not ActiveCell.Value = ""
    If Not ActiveCell(1, -1).Borders(xlEdgeRight).LineStyle = xlContinuous Then
        IntRBSValue = ActiveCell.Value
        IntRBSValue2 = ActiveCell(1, 2).Value
        ActiveCell.EntireRow.Delete
        Do While ActiveCell(1, -1).Borders(xlEdgeRight).LineStyle = xlContinuous
            ActiveCell(1, -1).Value = IntRBSValue
            ActiveCell(1, 0).Value = IntRBSValue2
            ActiveCell(2, 1).Select
        Loop
    End If
Loop

'Remove outdated no more used RBS entries prefixed with "XXX_"
Cells(2, 5).Select
Do While Not ActiveCell.Value = ""
    If Left(ActiveCell, 4) = "XXX_" Then
        ActiveCell.EntireRow.Delete
    Else: ActiveCell(2, 1).Select
    End If
Loop
'Remove not used columns
Columns("G:G").EntireColumn.Delete
Columns("H:P").EntireColumn.Delete
Cells(1, 1).Select

Else
    Exit Sub
End If

End Sub

Sub SmC_TimeCardActuals()
'Macro generated by Roland.Benz@Syngenta.com and franz.schuermann@syngenta.com (PMEC,
Project Management Excellence)
'Date: 26.9.2011
'Macro for Lee Hubbard
Dim Start As Date: Dim Duration As Date
Start = Now()

'MsgBox to activate the right sheet, &vbCrLf &
If 1 Then
    Response = MsgBox( _

```



```

"Please select the Excel sheet you want the macro to be applied to!" & Chr(13) & _
" " & Chr(13) & _
"Click Yes: if the right sheet is already selected" & Chr(13) & _
"      Otherwise" & Chr(13) & _
"Click No: and select the Excel sheet by clicking on its tab at the" & Chr(13) & _
"      bottom and then restart the macro", _
vbYesNo)
End If

'Execute the tasks
'Copy the input file ActiveSheet.Name in a new file Sh_Source and make changes
'Make new sheets Sh_Pivot and make the pivots
If Response = vbYes Then

    'Make a copy of the input sheet, change the sheet tab color and then make some some changes
    Dim Sh As String
    Sh = ActiveSheet.Name ' msgbox asks to make the right sheet active
    Dim Sh_Source As String
    Sh_Source = "ActualsByWeek"
    Dim Sh_Source2 As String
    Sh_Source2 = "RBS Table"

    If 1 Then
        Call z_ShNewFlatValueCopy(Sh, Sh_Source, "End")
        'change the colour of the sheet name on the tab
        With ActiveWorkbook.Sheets(Sh_Source).Tab
            .Color = RGB(0, 32, 90)
            .TintAndShade = 0
        End With
    End If

    If 1 Then
        'Add a column for person site information
        Sheets(Sh_Source).Select
        Rows("1:1").Find(What:="EMPLOYEE_NAME", LookAt:=xlWhole).Select
        ActiveCell.Offset(0, 1).EntireColumn.Insert
        ActiveCell(1, 2).Value = "SITE"
        'Fill the new column with person site info
        Range(ActiveCell(2, 1), ActiveCell(2, 1).End(xlDown)).Select
        For Each rSite In Selection.Cells
            rSite.Offset(0, 1).Value = Right(rSite.Value, 4)
        Next
    End If

    If 1 Then
        'Add a column for a concatenated string of PI ID and Task ID
        Sheets(Sh_Source).Select
        Rows("1:1").Find(What:="STAFF_DAYS", LookAt:=xlWhole).Select
        ActiveCell.Offset(0, 1).EntireColumn.Insert
        ActiveCell(1, 2).Value = "IDENTIFICATION"
        Range(ActiveCell(2, 2), ActiveCell(2, 1).End(xlDown).Offset(0, 1)).Select
        For Each rSite In Selection.Cells
            rSite.Value = rSite.Offset(0, -16).Value & "-" & rSite.Offset(0, -12).Value
        Next
        Range("A1").Select
    End If
End If

```

```

End If
If 1 Then
    'Add a column for Budget Group and map the Budget group values from Sh_Source2 to
Sh_Source
    'SmcSigma: SmCExtract: PMEC Term: Lee's term:
    'SmC Level 2->RBS Level 1->Resource Group->Budget Group
    'SmC Level 3->RBS Level 2->Resource
    'SmC Level 4->RBS Level 3->Resource Role -> Budget Center
    Sheets(Sh_Source).Select
    Rows("1:1").Find(What:="TIMECARD_COMMENT", LookAt:=xlWhole).Select
    ActiveCell.Offset(0, 1).EntireColumn.Insert
    ActiveCell(1, 2).Value = "BUDGET_GROUP"
    Call z_ShMapColumns(Sh_Source2, "RBS Level 3 Description", Array("RBS Level 1 Description"),
Sh_Source, "BUDGET_CENTER", Array("BUDGET_GROUP"), "Log_1")
End If

'Make new sheets and the pivots from Sh_Source
Dim Sh_Pivot As String
Dim RowU, ColL, RowD, ColR As Long
Dim Piv_RowD As Long
Dim Source_Rng As Range
Dim Piv_sULCell As String
Dim Piv_ULCell As Range
Dim Piv_F_R_C_V(0 To 3) As Variant
Dim PivName(0 To 2) As String

'Make a new sheet and add a pivot
'Name of the sheet
Sh_Pivot = "TimeToDateByTask"
If 1 Then
    Call z_ShNew(Sh_Pivot, "End")
    'change the colour of the sheet name on the tab
    With ActiveWorkbook.Sheets(Sh_Pivot).Tab
        .Color = RGB(255, 255, 0)
        .TintAndShade = 0
    End With
End If
If 1 Then
    'Parameters for the z_GeneratePivotTable function and its call
    'Name of the pivot sheet
    'Sh_Pivot = "Time-to-Date-By-Task-RB"
    'Name of the source sheet
    'Sh_Source = "ActualsByWeek-RB"
    'Determine the range of Sh_Source and create a range object
    RowU = 1: ColL = 1
    RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
    Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
    'Create a range object for the upper left corner of the pivot
    Piv_sULCell = "B9"
    Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
    'Determine the pivot fields
    Piv_F_R_C_V(0) = Array(13, 9, 11, 7, 12, 10) 'filter
    Piv_F_R_C_V(1) = Array(17, 5) 'row labels

```

```

Piv_F_R_C_V(2) = Array() 'column labels
Piv_F_R_C_V(3) = Array(16) 'values
'Call the function that creates the pivot
PivName(0) = z_GeneratePivotTable(Piv_ULCell, Source_Rng, Sh_Source, Sh_Pivot, Piv_F_R_C_V)
End If
If 1 Then
    'Pivot PivName(0)
    'set the filters
    Dim CurrentYear As String
    CurrentYear = Year(Date)
    ActiveSheet.PivotTables(PivName(0)).PivotFields("RESOURCE_YEAR").ClearAllFilters
    ActiveSheet.PivotTables(PivName(0)).PivotFields("RESOURCE_YEAR").CurrentPage = CurrentYear
    'change the column width of col A
    Columns("A:A").ColumnWidth = 5
    'insert formula into the Sh_Pivot
    Sheets(Sh_Pivot).Select
    Range("D7").Select
    Selection.Formula = "=GETPIVOTDATA("""STAFF_DAYS""",$B$9)"
    'format columns in Sh_Pivot
    Range("D2:D7").Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent3
        .TintAndShade = 0.599993896298105
        .PatternTintAndShade = 0
    End With
    Columns("A:A").ColumnWidth = 3
    Columns("D:D").Select
    Selection.NumberFormat = "#,##0.0"
    ActiveWorkbook.Save
    Range("B9").Select
End If

'Make a new sheet and add two pivots
'Name of the sheet
Sh_Pivot = "DataQualityLocation"
If 1 Then
    Call z_ShNew(Sh_Pivot, "End")
    'change the colour of the sheet name on the tab
    With ActiveWorkbook.Sheets(Sh_Pivot).Tab
        .Color = RGB(243, 130, 37)
        .TintAndShade = 0
    End With
End If
If 1 Then
    'Parameters for the z_GeneratePivotTable function and its call
    'Name of the pivot sheet
    'Sh_Pivot = "DataQualityLocation-RB"
    'Name of the source sheet
    'Sh_Source = "ActualsByWeek-RB"
    'Determine the range of Sh_Source and create a range object
    RowU = 1: Coll = 1

```

```

RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'Create a range object for the upper left corner of the pivot
Piv_sULCell = "B7"
Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
'Determine the pivot fields
Piv_F_R_C_V(0) = Array(13, 9) 'filter (enter numbers in proper order)
Piv_F_R_C_V(1) = Array(12, 19) 'row labels (enter numbers in reverse order)
Piv_F_R_C_V(2) = Array(7) 'column labels
Piv_F_R_C_V(3) = Array(16) 'values
'Call the function that creates the pivot
PivName(1) = z_GeneratePivotTable(Piv_ULCell, Source_Rng, Sh_Source, Sh_Pivot, Piv_F_R_C_V)
End If
If 1 Then
'Parameters for the z_GeneratePivotTable function and its call
'Name of the sheet
'Sh_Pivot = "DataQualityLocation-RB"
'Determine the range of Sh_Source and create a range object
RowU = 1: ColL = 1
RowD = z_RowSize(1, Sh_Source): ColR = z_ColSize(1, Sh_Source)
Set Source_Rng = Sheets(Sh_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))
'Create a range object for the upper left corner of the pivot
Piv_RowD = z_RowSize(2, Sh_Pivot)
Piv_sULCell = "B" & Piv_RowD + 8 'place the second pivot under the first pivot
Set Piv_ULCell = Sheets(Sh_Pivot).Range(Piv_sULCell)
'Determine the pivot fields
Piv_F_R_C_V(0) = Array(13, 9) 'filter (enter numbers in proper order)
Piv_F_R_C_V(1) = Array(17, 11, 19) 'row labels (enter numbers in reverse order)
Piv_F_R_C_V(2) = Array(7) 'column labels
Piv_F_R_C_V(3) = Array(16) 'values
'Call the function that creates the pivot
PivName(2) = z_GeneratePivotTable(Piv_ULCell, Source_Rng, Sh_Source, Sh_Pivot, Piv_F_R_C_V)
End If
'Add some formats and texts
If 1 Then
'Pivot PivName(1)
'set the filters
ActiveSheet.PivotTables(PivName(1)).PivotFields("RESOURCE_YEAR").ClearAllFilters
ActiveSheet.PivotTables(PivName(1)).PivotFields("RESOURCE_YEAR").CurrentPage = CurrentYear
ActiveSheet.PivotTables(PivName(1)).PivotFields("TASK_LOCATION").ClearAllFilters
ActiveSheet.PivotTables(PivName(1)).PivotFields("TASK_LOCATION").CurrentPage = "(blank)"
'change the column width of col A
Columns("A:A").ColumnWidth = 5
'add some text
Range("B2").Select
ActiveCell.Value = "Number of Tasks with recorded staff-days, Task Location =0"
Range("E2").Select
ActiveCell.Value = "241"
'aligment
Range("B2:D2").Select
With Selection
.HorizontalAlignment = xlLeft
.VerticalAlignment = xlBottom

```

```

.WrapText = False
.Orientation = 0
.AddIndent = False
.IndentLevel = 0
.ShrinkToFit = False
.ReadingOrder = xlContext
.MergeCells = False
End With
'merge cells
Range("B2:D2").Select
Selection.Merge
'change the font
Range("B2:E2").Select
Selection.Font.Bold = True
'add borders
Range("B2:E2").Select
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
With Selection.Borders(xlEdgeLeft)
.LineStyle = xlContinuous
.ColorIndex = 0
.TintAndShade = 0
.Weight = xlThick
End With
With Selection.Borders(xlEdgeTop)
.LineStyle = xlContinuous
.ColorIndex = 0
.TintAndShade = 0
.Weight = xlThick
End With
With Selection.Borders(xlEdgeBottom)
.LineStyle = xlContinuous
.ColorIndex = 0
.TintAndShade = 0
.Weight = xlThick
End With
With Selection.Borders(xlEdgeRight)
.LineStyle = xlContinuous
.ColorIndex = 0
.TintAndShade = 0
.Weight = xlThick
End With
With Selection.Borders(xlInsideVertical)
.LineStyle = xlContinuous
.ColorIndex = 0
.TintAndShade = 0
.Weight = xlThick
End With
With Selection.Borders(xlInsideHorizontal)
.LineStyle = xlContinuous
.ColorIndex = 0
.TintAndShade = 0
.Weight = xlThick

```

```

End With
'freeze panes
Rows("9:9").Select
ActiveWindow.FreezePanels = True
'Pivot PivName(2)
'copy/paste some text
Range("B2:E2").Select
Selection.Copy
Range("B" & Piv_RowD + 3).Select
ActiveSheet.Paste
'add filters
ActiveSheet.PivotTables(PivName(2)).PivotFields("RESOURCE_YEAR").ClearAllFilters
ActiveSheet.PivotTables(PivName(2)).PivotFields("RESOURCE_YEAR").CurrentPage = CurrentYear
ActiveSheet.PivotTables(PivName(2)).PivotFields("TASK_LOCATION").ClearAllFilters
ActiveSheet.PivotTables(PivName(2)).PivotFields("TASK_LOCATION").CurrentPage = "(blank)"
Range("B7").Select
End If
Else
Exit Sub
End If

```

```

'save the workbook
'ActiveWorkbook.Save
Duration = Now() - Start
Debug.Print Duration
End Sub

```

```

Private Function z_ShMapColumns(Sh_from As String, ColName_Key_from As String, ByRef
ColNames_from As Variant, _
    Sh_to As String, ColName_Key_to As String, ByRef ColNames_to As Variant, _
    Optional Sh_log As String, Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
Application.ScreenUpdating = False
'Start time measuring
Dim Start As Date: Dim Duration As Date
Start = Now()

```

```

'create a matrix with column names and indexes
MapMatrix = MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from,
ColNames_from, Sh_to, ColNames_to)

```

```

'Find the column index of the KeyName in "Sh_from"
Dim ColIndex_Key_from As Long
ColIndex_Key_from = z_GetColumnIndex(ColName_Key_from, 1, Sh_from)

```

```

'Find the column index of the KeyName in "Sh_to"
Dim ColIndex_Key_to As Long
ColIndex_Key_to = z_GetColumnIndex(ColName_Key_to, 1, Sh_to)

```

```

'Determine the row size in Sh_to
Sheets(Sh_to).Activate
RowSize_to = z_RowSize(ColIndex_Key_to, Sh_to)

```

```

'Select the range in the column Key_to
Sheets(Sh_to).Activate
Range(Cells(2, ColIndex_Key_to), Cells(RowSize_to, ColIndex_Key_to)).Select

'Iterate through the rows with "rcheck" = Pidentifier
Dim ilog As Long
ilog = 2
For Each ValueInCol_Key_to In Selection.Cells
    'if ValueInCol_Key_to is found in Sh_from then perform the mapping
    If Not Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to,
LookAt:=xlWhole) Is Nothing Then
        'iterate through the columns with the help of the MapMatrix
        For j = LBound(MapMatrix) To UBound(MapMatrix) Step 1
            'read the indices
            ColIndex_from_j = MapMatrix(j, 1)
            ColIndex_to_j = MapMatrix(j, 3)
            'map
            ValueInCol_Key_to.Offset(0, (ColIndex_to_j) - ColIndex_Key_to).Value = _
                Sheets(Sh_from).Columns(ColIndex_Key_from).Find(What:=ValueInCol_Key_to, _
                LookAt:=xlWhole).Offset(0, (ColIndex_from_j) - ColIndex_Key_from)
        Next j
    Else
        'write not found ValueInCol_Key_to in Sh_from into Sh_log
        On Error Resume Next
        Sheets(Sh_log).Cells(ilog, 2) = ValueInCol_Key_to
        Sheets(Sh_log).Cells(ilog, 4) = " not found, map them from another source file Sh_from"
        ilog = ilog + 1
        On Error GoTo 0
    End If
Next

'In case the mapping has changed the row height
Sheets(Sh_to).Activate
Cells.Select
Selection.Rows.RowHeight = 15
Application.ScreenUpdating = True
'Write the durations into the logfile
On Error Resume Next
Duration = Now() - Start
Sheets(Sh_log).Cells(ilog + 2, 1) = "Duration" & CStr(Duration)
On Error GoTo 0
End Function

```

```

Private Function z_ChkColExistence(Sh As String, ByRef ColNames As Variant, Sh_log As String) As
Boolean

```

```

    'The column existence check is assumed to find all column names at the beginning
    z_ChkColExistence = True
    'Determine the column indices of the ColNames array in Sh
    Dim ColName_i As String
    ReDim Matrix_ColNameColIndex(0 To UBound(ColNames), 0 To 1) As Variant
    'iterate through the array
    For i = LBound(ColNames_from) To UBound(ColNames) Step 1
        ColName_i = CStr(ColNames(i))
    
```

```

Matrix_ColNameColIndex(i, 0) = ColName_i
Matrix_ColNameColIndex(i, 1) = z_GetColumnIndex(ColName_i, 1, Sh)
Debug.Print Matrix_ColNameColIndex(i, 0) & " " & Matrix_ColNameColIndex(i, 1)
If Matrix_ColNameColIndex(i, 1) = 0 Then
    'write errors into the logfile
    Sheets(Sh_log).Cells(iLog, 2) = "ColName: "
    Sheets(Sh_log).Cells(iLog, 3) = Matrix_ColNameColIndex(i, 0)
    Sheets(Sh_log).Cells(iLog, 4) = "not found in " & Sh
    iLog = iLog + 1
    'The column existence check has detected an unfound column name
    z_ChkColExistence = False
Next i
End Function

Private Function MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex(Sh_from As String,
ByRef ColNames_from As Variant, _
    Sh_to As String, ByRef ColNames_to As Variant) As Variant
    'Determine the column indizes in Sh and Sh_new,
    Dim ColName_from_i As String
    Dim ColName_to_i As String
    ReDim Matrix_ColName1Index1_ColName2Index2(0 To UBound(ColNames_from), 0 To 3) As
Variant

    For i = LBound(ColNames_from) To UBound(ColNames_from) Step 1
        ColName_from_i = CStr(ColNames_from(i))
        Matrix_ColName1Index1_ColName2Index2(i, 0) = ColName_from_i
        Matrix_ColName1Index1_ColName2Index2(i, 1) = z_GetColumnIndex(ColName_from_i, 1,
Sh_from)
        ColName_to_i = CStr(ColNames_to(i))
        Matrix_ColName1Index1_ColName2Index2(i, 2) = ColName_to_i
        Matrix_ColName1Index1_ColName2Index2(i, 3) = z_GetColumnIndex(ColName_to_i, 1, Sh_to)
        Debug.Print Matrix_ColName1Index1_ColName2Index2(i, 0) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 1) _
            & " " & Matrix_ColName1Index1_ColName2Index2(i, 2) & " " &
Matrix_ColName1Index1_ColName2Index2(i, 3)
    Next i
    MakeMatrix_Shfrom_ColNameColIndex_Shto_ColNameColIndex =
Matrix_ColName1Index1_ColName2Index2
End Function

Private Function z_ShNewFlatValueCopy(Sh As String, Sh_new As String, Optional Where As String,
Optional ByRef Sh_Ref As Worksheet)
    'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    'Date: 26.9.2011
    'Input: Name of the Sh to copy, Name of the new Sh_new, where to place the new Sh_new,
    ' before or after some Sh_Ref, at the begin or the end
    Call z_ShAdd(Where, Sh_Ref)
    On Error Resume Next
    ActiveSheet.Name = Sh_new
    If Err.Number <> 0 Then
        Application.DisplayAlerts = False
        ActiveSheet.Delete
        Application.DisplayAlerts = True
    End If
End Function

```



```

        Sheets(Sh_new).Cells.ClearContents
    End If
    On Error GoTo 0
    Sheets(Sh).Cells.Copy
    Sheets(Sh_new).Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Sheets(Sh_new).Columns("A:GA").ColumnWidth = 20
End Function

```

```

Private Function z_ShNew(Sh As String, Optional Where As String, Optional ByRef Sh_Ref As
Worksheet, Optional ByRef Wb As Workbook)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Name of the new Sh, where to place the new Sh, before or after some Sh_Ref, at the begin or
the end
On Error Resume Next
WorksheetExists = (Sheets(Sh).Name <> "")
On Error GoTo 0
If WorksheetExists = False Then
    Call z_ShAdd(Where, Sh_Ref)
    ActiveSheet.Name = Sh 'Worksheets.Add(Before:=Worksheets(1)).Name = Sh
End If
'Clear contents
Sheets(Sh).Activate
ActiveSheet.Cells.Select
Selection.ClearContents
'Format
Sheets(Sh).Columns.ColumnWidth = 20
Sheets(Sh).Rows.RowHeight = 15
End Function

```

```

Private Function z_ShAdd(Optional Where As String, Optional ByRef Sh_Ref As Worksheet)
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: where to place the added Sh, before or after some Sh_Ref, at the begin or the end
If Not Sh_Ref Is Nothing Then
    If Where = ("Before:=") Then
        Sheets.Add Sh_Ref
    ElseIf Where = ("After:=") Then
        Sheets.Add , Sh_Ref
    Else
        Sheets.Add Before:=Sheets(1)
    End If
Else
    If Where = ("End") Then
        Sheets.Add After:=Sheets(Sheets.Count)
    ElseIf Where = ("Begin") Then
        Sheets.Add Before:=Sheets(1)
    Else
        Sheets.Add Before:=Sheets(1)
    End If
End If

```

End Function

Private Function z\_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef MyWb As Workbook) As Long

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: column, Output: row with the last entry in that column

'SearchCol datatype changed from integer

'Activate the Sheet

Sheets(Sh).Activate

'Determine the row size

z\_RowSize = If(IsEmpty(Cells(1048576, SearchCol)), Cells(1048576, SearchCol).End(xlUp).Row, 1048576)

End Function

Private Function z\_ColSize(SearchRow As Long, Optional Sh As String, Optional ByRef MyWb As Workbook) As Long

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input: Row, Output: column with the last entry in that row

'SearchCol datatype changed from integer

'Activate the Sheet

Sheets(Sh).Activate

'Determine the col size

z\_ColSize = If(IsEmpty(Cells(SearchRow, 16384)), Cells(SearchRow, 16384).End(xlToLeft).Column, 16384)

End Function

Private Function z\_GeneratePivotTable(Piv\_ULCell As Range, Source\_Rng As Range, \_  
Sh\_Source As String, Sh\_Pivot As String, Piv\_F\_R\_C\_V As Variant) As String

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Date: 26.9.2011

'Input parameters (arguments) for the z\_GeneratePivotTable function and its call

'Name of the pivot sheet

'Sh\_Pivot = "TimeToDateByTask"

'Name of the source sheet

'Sh\_Source = "ActualsByWeek"

'Determine the range of Sh\_Source and create a range object

'RowU = 1: ColL = 1

'RowD = z\_RowSize(1, Sh\_Source): ColR = z\_ColSize(1, Sh\_Source)

'Set Source\_Rng = Sheets(Sh\_Source).Range(Cells(RowU, ColL), Cells(RowD, ColR))

'Create a range object for the upper left corner of the pivot

'Piv\_sULCell = "B9"

'Set Piv\_ULCell = Sheets(Sh\_Pivot).Range(Piv\_sULCell)

'Determine the pivot fields

'Piv\_F\_R\_C\_V(0) = Array(13, 9, 11, 7, 12, 10) 'filter

'Piv\_F\_R\_C\_V(1) = Array(17, 5) 'row labels

'Piv\_F\_R\_C\_V(2) = Array() 'column labels

'Piv\_F\_R\_C\_V(3) = Array(16) 'values

'Creat the strings for the function ActiveWorkbook.PivotCaches.Create() further below

Dim sSource\_Rng As String

sSource\_Rng = Sh\_Source & "!" & Source\_Rng.Address(ReferenceStyle:=xlR1C1)

```

Dim sPivot_Rng As String
sPivot_Rng = Sh_Pivot & "!" & Piv_ULCell.Address(ReferenceStyle:=xlR1C1)

'Store the range.address information into an array
Dim RngAddress As Variant
RngAddress = z_RangeAddressAsArray(Source_Rng)

'Store the Column names into an array
ReDim PivChosenField(RngAddress(2) To RngAddress(4)) As String
For i = RngAddress(2) To RngAddress(4)
    PivChosenField(i) = Source_Rng.Cells(1, i)
Next i

'Store the column names of the ReportFilter, RowLabel, ColLabel and Value into arrays
ReDim PivReportFilter(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivRowLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivColLabel(RngAddress(2) - 1 To RngAddress(4) - 1) As String
ReDim PivValue(RngAddress(2) - 1 To RngAddress(4) - 1) As String
For i = RngAddress(2) - 1 To RngAddress(4) - 1
    On Error Resume Next
    PivReportFilter(i) = PivChosenField(Piv_F_R_C_V(0)(i))
    On Error GoTo 0
    On Error Resume Next
    PivRowLabel(i) = PivChosenField(Piv_F_R_C_V(1)(i))
    On Error GoTo 0
    On Error Resume Next
    PivColLabel(i) = PivChosenField(Piv_F_R_C_V(2)(i))
    On Error GoTo 0
    On Error Resume Next
    PivValue(i) = PivChosenField(Piv_F_R_C_V(3)(i))
    On Error GoTo 0
Next i

'generate Pivot
Sheets(Sh_Pivot).Select
Piv_ULCell.Select
ActiveWorkbook.PivotCaches.Create( _
    SourceType:=xlDatabase, _
    SourceData:=sSource_Rng, _
    Version:=xlPivotTableVersion12).CreatePivotTable _
    TableDestination:=sPivot_Rng, _
    TableName:=Piv_Name, _
    DefaultVersion:=xlPivotTableVersion12

'get Pivot table name
'if PivName = "PivotTable" is used more than once an iteger is added to the name
PivName = ActiveSheet.PivotTables(1).Name

For i = RngAddress(2) - 1 To RngAddress(4) - 1
    'Define row labels
    On Error Resume Next
    With ActiveSheet.PivotTables(PivName).PivotFields(PivRowLabel(i))
        .Orientation = xlRowField
    End With

```

```

        .Position = 1
    End With
    On Error GoTo 0
    'Define column labels
    On Error Resume Next
    With ActiveSheet.PivotTables(PivName).PivotFields(PivCollLabel(i))
        .Orientation = xlColumnField
        .Position = 1
    End With
    On Error GoTo 0
    'Define values
    On Error Resume Next
    ActiveSheet.PivotTables(PivName).AddDataField ActiveSheet.PivotTables( _
    PivName).PivotFields(PivValue(i)), "Sum of STAFF_DAYS", xlSum
    On Error GoTo 0
    'Define report filters
    On Error Resume Next
    With ActiveSheet.PivotTables(PivName).PivotFields(PivReportFilter(i))
        .Orientation = xlPageField
        .Position = 1
    End With
    On Error GoTo 0
Next i

'Change the layout
With ActiveSheet.PivotTables(PivName)
    .InGridDropZones = True
    .RowAxisLayout xlTabularRow
End With

'Define all columns from : Choose fields to add to report
For i = RngAddress(2) To RngAddress(4)
    ActiveSheet.PivotTables(PivName).PivotFields(PivChosenField(i)).Subtotals = _
    Array(False, False, False, False, False, False, False, False, False, False, False)
Next i

'output
z_GeneratePivotTable = PivName

End Function

Private Function z_RangeAddressAsArray(Rng As Range) As Variant
'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
'Date: 26.9.2011
'Input: Range(Cells(a,b),Cells(c,d)), Output: Array(a,b,c,d)
sRngAddress = Rng.Address(ReferenceStyle:=xlR1C1)
DltrLst = Array("$", "R", "C", ":")
RplLst = Array("@", "@", "@", "@")
For k = LBound(DltrLst) To UBound(DltrLst)
    If RplLst(k) = Empty Then
        l = RplLst(0)
    Else
        l = k
    End If
Next k

```

```

End If
sRngAddress = Replace(sRngAddress, Dltrlst(k), RplLst(l))
Next k
sRngAddress = Replace(sRngAddress, RplLst(0) & RplLst(0), RplLst(l))
Dim RngAddress As Variant
RngAddress = Split(sRngAddress, RplLst(0))
For i = 1 To 4 Step 1
    RngAddress(i - 1) = RngAddress(i)
Next i
ReDim Preserve RngAddress(1 To 4)
z_RangeAddressAsArray = RngAddress

```

End Function

Private Function z\_GetColumnIndex(ByRef SearchString As String, SearchRow As Integer, Optional Sh As String, Optional ByRef Wb As Workbook) As Long

'Macro generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Output datatype change from Variant

```

Dim CellIndexStr As String 'In R1C1 Format
Dim CellIndexArr() As String 'Splited R1C1 Format
Dim ColIndex As Integer

```

```

'Activate the right Wb and Sh
On Error GoTo OptionalArgument:
Wb.Activate
On Error GoTo 0
Sheets(Sh).Activate

```

```

'find column name
On Error GoTo NameExpectedNotExistent:
Rows(SearchRow).Find(What:=SearchString, LookAt:=xlWhole).Select
On Error GoTo 0
'find column index
CellIndexStr = ActiveCell.Address(ReferenceStyle:=xlR1C1)
CellIndexArr = Split(CellIndexStr, "C")
ColIndex = CInt(CellIndexArr(1))
'Output
z_GetColumnIndex = ColIndex
Exit Function

```

```

OptionalArgument:
Resume Next

```

```

NameExpectedNotExistent:
ColIndex = 0
z_GetColumnIndex = ColIndex

```

End Function

## File: To Roland task 1

Sub ColumnWithNames\_2\_EmailList()

'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

'Split the string

Dim DelimiterList As Variant

Dim ReplacementList As Variant

Dim StringSplit\_Dim2 As Variant

Dim Input\_Range As Range

Dim Output\_Range As Range

Dim Input\_Indices As Variant

Dim Output\_Indices As Variant

DelimiterList = Array("#", ";")

ReplacementList = Array("@", "@")

'input

Set Input\_Range = Application.InputBox(prompt:="Select the input range (cells or column) and click OK", Type:=8)

'get the indices of the input range

Input\_Indices = z\_RangeToIndices(Input\_Range)

'output

Set Output\_Range = Application.InputBox(prompt:="Select the output range (cells or column) and click OK", Type:=8)

'get the indices of the output range

Output\_Indices = z\_RangeToIndices(Output\_Range)

If 1 Then

StringSplit\_Dim2 = z\_StringSplit(DelimiterList, ReplacementList, Input\_Indices, Output\_Indices)

End If

End Sub

Function z\_StringSplit(Dltrlst As Variant, Rpllst As Variant, Indices\_from As Variant, Indices\_to As Variant, \_

Optional Sh As String, Optional ByVal Wb As Workbook) As Variant

'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)

If Sh <> "" Then

Sheets(Sh).Activate

End If

'Copy/paste the range with indices\_from to the range with indices\_to

Range(Cells(Indices\_from(0), Indices\_from(1)), Cells(Indices\_from(2), Indices\_from(3))).Select

Selection.Copy Destination:=Cells(Indices\_to(0), Indices\_to(1))

'Select the range of the copied cells and make the replacements

Dim Rng\_to As Range

If Indices\_from(3) - Indices\_from(1) = 0 Then

Set Rng\_to = Range(Cells(Indices\_to(0), Indices\_to(1)), Cells(Indices\_from(2) - \_  
Indices\_from(0) + 1, Indices\_to(1)))

Rng\_to.Select

Else

Set Rng\_to = Range(Cells(Indices\_to(0), Indices\_to(1)), Cells(Indices\_to(0), \_  
Indices\_from(3) - Indices\_from(1) + 1))

```

    Rng_to.Select
End If
For k = LBound(Dltrlst) To UBound(Dltrlst)
    If RplLst(k) = Empty Then
        l = RplLst(0)
    Else
        l = k
    End If
    Selection.Replace What:=Dltrlst(k), Replacement:=RplLst(l), LookAt:=xlPart, _
        SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
        ReplaceFormat:=False
Next k

```

```

'Write the strings of all cells into one cell with indices_to
Dim sAllCellValues As String
For Each cll In Selection.Cells
    If cll <> "" Then
        sAllCellValues = sAllCellValues & "@@" & cll.Value
    Else
        End If
Next cll

```

```

'clear the range with indices_to
Selection.ClearContents

```

```

'Read the data into an array splitted with a delimiter
Dim sSplitArray As Variant
sSplitArray = Split(sAllCellValues, "@")

```

```

'remove double entries
For iter = 2 To UBound(sSplitArray) Step 4
    For Iter2 = iter + 4 To UBound(sSplitArray) Step 4
        If sSplitArray(iter) = sSplitArray(Iter2) Then
            sSplitArray(iter) = "DoubleEntryToDelete"
        End If
    Next Iter2
Next iter

```

```

'create a new string with the data of interest
Dim sSplitOfInterest As String
For iter = 2 To UBound(sSplitArray) Step 4
    If sSplitArray(iter) <> "DoubleEntryToDelete" Then
        sSplitOfInterest = sSplitOfInterest & sSplitArray(iter) & "; "
    End If
Next iter

```

```

'write out the results
Dim rowH As Integer
'rowH = Cells(Indices_to(0), Indices_to(1)).RowHeight
'ColW = Cells(Indices_to(0), Indices_to(1)).ColumnWidth
Cells(Indices_to(0), Indices_to(1)).Select
Cells(Indices_to(0), Indices_to(1)) = sSplitOfInterest
Selection.ColumnWidth = 40

```

```
Selection.EntireRow.AutoFit
End Function
```

```
Function z_RowSize(SearchCol As Long, Optional Sh As String, Optional ByRef Wb As Workbook) As Long
```

```
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

```
'Date: 26.9.2011
```

```
'Input: column, Output: row with the last entry in that column
```

```
'SearchCol datatype changed from integer
```

```
'Activate the Sheet
```

```
Sheets(Sh).Activate
```

```
'Determine the row size
```

```
z_RowSize = If(IsEmpty(Cells(1048576, SearchCol)), Cells(1048576, SearchCol).End(xlUp).Row, 1048576)
```

```
End Function
```

```
Function z_sCellToIndex(ByRef CellIndexStr As String) As Variant
```

```
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

```
Dim CellIndexArr() As String 'Splited R1C1 Format
```

```
Dim ColIndex As Integer
```

```
Dim RowIndex As Integer
```

```
Dim CellIndices(0 To 1) As Long
```

```
'find column index
```

```
CellIndexArr = Split(CellIndexStr, "C")
```

```
ColIndex = CInt(CellIndexArr(1))
```

```
CellIndexArr = Split(CellIndexArr(0), "R")
```

```
RowIndex = CInt(CellIndexArr(1))
```

```
CellIndices(0) = RowIndex
```

```
CellIndices(1) = ColIndex
```

```
'Output
```

```
z_sCellToIndex = CellIndices
```

```
End Function
```

```
Function z_RangeToIndices(ByRef Rng As Range) As Variant
```

```
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
```

```
Dim RangeIndices(0 To 3) As Long
```

```
Dim CellsArray() As String
```

```
Dim sAddr As String
```

```
sAddr = Rng.Address(ReferenceStyle:=xlR1C1)
```

```
CellsArray = Split(sAddr, ":")
```

```
Dim CellIndicesUL() As Long
```

```
On Error GoTo RangelsColumnOrRow
```

```
CellIndicesUL = z_sCellToIndex(CellsArray(0))
```

```
On Error GoTo 0
```

```
Dim CellIndicesLR() As Long
```

```
On Error GoTo RangelsCell
```

```
CellIndicesLR = z_sCellToIndex(CellsArray(1))
```

```
On Error GoTo 0
```

```
RangeIndices(0) = CellIndicesUL(0)
```



```

RangeIndices(1) = CellIndicesUL(1)
RangeIndices(2) = CellIndicesLR(0)
RangeIndices(3) = CellIndicesLR(1)

z_RangeToIndices = RangeIndices
Exit Function
RangeCell:
CellIndicesLR = z_sCellToIndex(CellsArray(0))
Resume Next
RangeColumnOrRow:
Dim RorC As String
RorC = Left(sAddr, 1)
OneOrMore = InStr(1, sAddr, ":", vbTextCompare)
'only one row or column
If OneOrMore = 0 Then
    If RorC = "C" Then
        RangeIndices(0) = 1
        RangeIndices(1) = z_sColumnToIndex(CellsArray(0))
        RangeIndices(2) = 1048534
        RangeIndices(3) = RangeIndices(0)
    ElseIf RorC = "R" Then
        RangeIndices(0) = z_sRowToIndex(CellsArray(0))
        RangeIndices(1) = 1
        RangeIndices(2) = RangeIndices(0)
        RangeIndices(3) = 16383
    Else
        Stop
    End If
'more than one row or column
Else
    If RorC = "C" Then
        RangeIndices(0) = 1
        RangeIndices(1) = z_sColumnToIndex(CellsArray(0))
        RangeIndices(2) = 1048534
        RangeIndices(3) = z_sColumnToIndex(CellsArray(1))
    ElseIf RorC = "R" Then
        RangeIndices(0) = z_sRowToIndex(CellsArray(0))
        RangeIndices(1) = 1
        RangeIndices(2) = z_sRowToIndex(CellsArray(1))
        RangeIndices(3) = 16383
    Else
        Stop
    End If
End If
z_RangeToIndices = RangeIndices
End Function

Function z_sColumnToIndex(ByRef ColIndexStrLeft As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim ColArray() As String
    ColArray = Split(ColIndexStrLeft, "C")
    z_sColumnToIndex = ColArray(1)
End Function

```

```

Function z_sRowToIndex(ByRef RowIndexStrUp As String) As Variant
'Procedure generated by Roland.Benz@Syngenta.com (PMEC, Project Management Excellence)
    Dim RowArray() As String
    RowArray = Split(RowIndexStrUp, "R")
    z_sRowToIndex = RowArray(1)
End Function

```

## **File: VBA Testmappe**

```

Private Sub Workbook_SheetChange(ByVal Sh As Object, _
    ByVal Source As Range)
    ' runs when a sheet is changed
    'Stop
End Sub

```

```

Private Sub Workbook_SheetActivate(ByVal Sh As Object)
    ' runs when a sheet is changed
    'Stop
End Sub

```

```

Private Sub Workbook_Open()
    'Stop
End Sub

```

```

Private Sub Workbook_WindowActivate(ByVal Wn As Window)
    'Stop
End Sub

```

```

Sub S1()
Dim i, PI1, WS, PosPI1, SumTK, PosWS, SumWS, TotalWS As Integer
Range(Cells(2, 4), Cells(100, 4)).Select: Selection.ClearContents
x = 2: y = 4
i = 2
Do Until Cells(i, 1) = ""
    If Cells(i, 1) = "PI" Then
        SumWS = 0
        PosWS = 0
        PI1 = Cells(i, x)
        PosPI1 = i
    ElseIf Cells(i, 1) = "WS" Then
        WS1 = 0
        WS = Cells(i, x)
        PosWS = i
        a = i + 1 'Wird nicht gebraucht!!!!
        If Cells(i + 1, 1) <> "TK" Then 'Eingefügt, Fall wenn WK ohne TK
            SumWS = SumWS + WS
        End If
    ElseIf Cells(i, 1) = "TK" Then
        a = i
    End If
    i = i + 1
Loop

```

```

Do Until Cells(a, 1) <> "TK"
    SumTK = SumTK + Cells(a, x)
    'WS = SumTK 'Fehler!!!!
    a = a + 1
    i = a - 1
Loop
If PosWS > 0 Then 'eingefügt, Fall wo Kosten auf WS
    WS1 = WS - SumTK
End If
Else 'eingefügt wegen error handling!!!!
    Stop
End If 'Fehler, von unten raufgenommen!!!!

If PosWS > 0 Then 'Fall TK nicht direkt unter PI
    Cells(PosWS, y) = WS - SumTK
End If
SumWS = SumWS + SumTK + WS1

If PosWS = 0 Then 'Fall TK direkt unter PI
    Cells(PosPI1, y) = WS - SumTK 'Fehler, wird mit nächster Zeile unbenutzt wieder
    überschrieben!!!!
End If
Cells(PosPI1, y) = PI1 - SumWS
If SumWS > PI1 Then 'Dieser Fall sollte nie eintreffen!!!!
    Cells(PosPI1, y) = PI1 - SumWS
End If
SumTK = 0
'End If 'Fehler, oben eingefügt!!!!
i = i + 1
Loop
End Sub

Dim PI_Array() As Variant
Dim PI_Array_Index() As Integer
Dim PI_Missing() As Variant
Dim PI_Missing_Index() As Integer
Option Explicit

Sub S10()
    Sheets("Tabelle10").Activate
    Dim i1, i2, i3, RowSize As Integer: i2 = 0: i3 = 0
    RowSize = If(IsEmpty(Range("A1048576")), Range("A1048576").End(xlUp).Row, 1048576)
    For i1 = 2 To RowSize
        If Cells(i1, 1) = "PI" Then
            i2 = i2 + 1
        Else
            i3 = i3 + 1
        End If
    Next i1
    ReDim Preserve PI_Array(0 To i2 - 1)
    ReDim Preserve PI_Array_Index(0 To i2 - 1)
    ReDim Preserve PI_Missing(0 To i3 - 1)

```

```
ReDim Preserve PI_Missing_Index(0 To i3 - 1)
```

```
i2 = 0: i3 = 0
```

```
For i1 = 2 To RowSize
```

```
    If Cells(i1, 1) = "PI" Then
```

```
        PI_Array(i2) = Cells(i1, 2)
```

```
        PI_Array_Index(i2) = i1
```

```
        i2 = i2 + 1
```

```
    Else
```

```
        PI_Missing(i3) = Cells(i1, 2)
```

```
        PI_Missing_Index(i3) = i1
```

```
        i3 = i3 + 1
```

```
    End If
```

```
Next i1
```

```
Call PrintOutArrays
```

```
End Sub
```

```
Sub PrintOutArrays()
```

```
    Dim j, i
```

```
    j = 2
```

```
    For i = LBound(PI_Array) To UBound(PI_Array)
```

```
        Cells(j, 5) = PI_Array(i)
```

```
        Cells(j, 6) = PI_Array_Index(i)
```

```
        j = j + 1
```

```
    Next i
```

```
    j = 2
```

```
    For i = LBound(PI_Missing) To UBound(PI_Missing)
```

```
        Cells(j, 7) = PI_Missing(i)
```

```
        Cells(j, 8) = PI_Missing_Index(i)
```

```
        j = j + 1
```

```
    Next i
```

```
End Sub
```

```
Sub S2()
```

```
    Dim Key As Range
```

```
    Dim wks As Worksheet: Set wks = Sheets("Tabelle2")
```

```
    RowSize = If(IsEmpty(wks.Range("A1000")), wks.Range("A1000").End(xlUp).Row, 1000)
```

```
    wks.Range(Cells(2, 2), Cells(RowSize, 2)).ClearContents
```

```
    For Each Key In wks.Range(Cells(2, 1), Cells(RowSize, 1))
```

```
        If Not wks.Columns("D:D").Find(What:=Key, LookAt:=xlWhole) Is Nothing Then
```

```
            Key.Offset(0, 1).Value = wks.Columns("D:D").Find(What:=Key, LookAt:=xlWhole).Offset(0, 1).Value
```

```
        End If
```

```
    Next Key
```

```
End Sub
```

```
Sub S3()
```

```
    Sheets("Tabelle3").Activate
```

```
    Rows(1).Select
```

```
    Selection.RowHeight = 35
```

```
    Rows("2:3").Select
```

Selection.RowHeight = 25

Range(Cells(1, 1), Cells(1, 5)).Select  
Selection.BorderAround ColorIndex:=1, Weight:=xlThick  
Selection.Interior.ColorIndex = 30  
Selection.Font.Name = "Arial"  
Selection.Font.Bold = True  
Selection.Font.Color = RGB(20, 255, 80)

Range(Cells(2, 1), Cells(3, 5)).Select  
Selection.BorderAround ColorIndex:=1, Weight:=xlThick  
Selection.Interior.ColorIndex = 4  
Selection.Font.ColorIndex = 31

Range(Cells(1, 1), Cells(13, 5)).Select  
Selection.Borders(11).Weight = xlThin  
Selection.Borders(12).Weight = xlThin  
Selection.BorderAround Weight:=xlThick  
Selection.HorizontalAlignment = xlCenter  
Selection.VerticalAlignment = xlBottom

Columns("A:E").Select  
Selection.EntireColumn.AutoFit

Range("C14").Select  
ActiveCell.FormulaR1C1 = "=SUM(R[-12]C:R[-1]C)"  
Selection.NumberFormat = "#,##0.0"  
Selection.ClearContents

Range("E14").Select  
ActiveCell.FormulaR1C1 = "=COUNT(RC[-2]:RC[-1])"  
Selection.NumberFormat = "#,##0.0"  
Selection.ClearContents

Cells.Select  
'Selection.Delete Shift:=xlUp  
'Selection.ClearContents  
Selection.ClearFormats  
Selection.RowHeight = 15  
Selection.ColumnWidth = 10.71

Cells(1, 1).Select

End Sub

Sub S4()  
Sheets("Tabelle4").Activate  
Cells(1, 1).Select  
Selection.Replace What:=" ", Replacement:="", LookAt:=xlPart  
Selection.TextToColumns \_  
    Destination:=Cells(1, 2), \_  
    DataType:=xlDelimited, \_

```

ConsecutiveDelimiter:=True, _
Comma:=True
Dim MyArray() As String
Dim j: j = 2
MyArray = Split(Cells(1, 1), ",")
For Each i In MyArray
    Cells(2, j) = i: j = j + 1
Next i
Range(Cells(1, 2), Cells(2, 50)).Select
Selection.ClearContents
End Sub

```

```

Sub S5()
Sheets("Tabelle5").Activate
Dim myRange As Range

```

```

Set myRange = Worksheets("Tabelle5").Range("A2:C21")
myRange.Select

```

```

answer = Application.WorksheetFunction.Min(myRange)
answer = Application.WorksheetFunction.Average(myRange)
answer = Application.WorksheetFunction.Sum(myRange)

```

```

End Sub

```

```

Sub S6()
Sheets("Tabelle6").Activate
Dim myRange As Range
Set myRange = Worksheets("Tabelle6").Range("A2:C21")
myRange.Copy Destination:=Range("F2:H21")
myRange.Sort Key1:=Cells(1, 1), Key2:=Cells(1, 2)
Range("F2:H21").Copy Destination:=Range("A2:A21")
End Sub

```

```

Sub S7()
Sheets("Tabelle7").Activate
Dim i As Integer: i = 3
Do Until Cells(i, 1) = ""
    Cells(i, 1).Select
    ActiveCell.EntireRow.Insert
    i = i + 2
Loop
End Sub

```

```

Sub S7_1()
For x = 40 To 1 Step -1
    If Cells(x, 1) = "" Then
        Cells(x, 1).Select
        ActiveCell.EntireRow.Delete
    End If
Next x
End Sub

```

```

Sub S8()
    Sheets("Tabelle8").Activate

    Cells.Select

    Cells(1, 1).Select: ActiveCell.EntireRow.Select
    Cells(1, 1).Select: ActiveCell.EntireColumn.Select

    Range("A2:C5").Select
    Range(Cells(2, 1), Cells(7, 3)).Select

    Rows(3).Select
    Rows("3:4").Select
    myRange = CStr(5) & ":" & CStr(9): Rows(myRange).Select

    Columns(1).Select
    Columns("A:B").Select

    Set mc = Worksheets(1).Cells(1, 1)
    Var = mc.AddressLocal(ReferenceStyle:=xlR1C1)
    Var = mc.AddressLocal(ReferenceStyle:=xlR1C1, _
        RowAbsolute:=False, _
        ColumnAbsolute:=False, _
        RelativeTo:=Worksheets(1).Cells(3, 3))    ' Z(-2)S(-2)

    myRange = "R" & "2" & "C" & "1" & ":" & "R" & "3" & "C" & "3"
    'inputRange = "R2C1:R3C3"
    inputRangeC = Application.ConvertFormula( _
        Formula:=myRange, _
        fromReferenceStyle:=xlR1C1, _
        toReferenceStyle:=xlA1)
    Range(myRangeC).Select

    myFormula = "=SUM(R2C1:R3C3)"
    myFormulaC = Application.ConvertFormula( _
        Formula:=myFormula, _
        fromReferenceStyle:=xlR1C1, _
        toReferenceStyle:=xlA1)
    Cells(3, 4) = myFormulaC

End Sub

Sub S9_1()
    Application.OnTime Now + TimeValue("00:00:15"), "my_Procedure"
End Sub

Sub S9_2()
    Application.OnTime TimeValue("13:26:30"), "my_Procedure"
End Sub
Sub my_Procedure()
    MsgBox "Hi"

```

```

End Sub
Sub S9_3()
If Application.Wait(Now + TimeValue("0:00:10")) Then
    MsgBox "Time expired"
End If
End Sub
Sub S9_4()
Application.SendKeys ("%fx")
End Sub
Sub S9_5()
Worksheets("Tabelle9").Activate
For i = 1 To Sheets.Count
    Cells(i, 1).Value = Sheets(i).Name
Next i
i = 1
For Each WS In Worksheets
    Cells(i, 2).Value = WS.Name
    i = i + 1
Next WS
'Set NewSheet = Sheets.Add(Type:=xlWorksheet)
'For i = 1 To Sheets.Count
'NewSheet.Cells(i, 1).Value = Sheets(i).Name
'Next i
End Sub
Sub S9_6()
Application.Speech.Speak "Hello"
End Sub
Sub S9_7()
a = ThisWorkbook.Path
b = Application.UserName
c = Application.Version
d = Application.OperatingSystem
e = ActiveWindow.Top
f = ActiveWindow.Left
g = ActiveWindow.Height
h = ActiveWindow.Width
End Sub
Sub S9_8()
For Each w In Workbooks
    If w.Name <> ThisWorkbook.Name Then
        w.Close savechanges:=True
    End If
Next w
End Sub

```