vbArc > Code Printer > CodePrinter.xlsm

--- Table of Contents: ---

14. CollectionToArray
15. ColorWords
16. ColorizeBlockLinksByLevel
17. Combine2Array
18. CommentsToOwnLine
19. ComponentTypeToString
20. CountOfCharacters
21. DPH
22. DebugPrintHairetu
23. DeleteWorksheet
24. EndTimer
25. FindAll
26. FolderExists
27. FoldersCreate
28. FormatPrinterTitles
29. FormatTextColors
30. GetCodeOf
31. GetCompText
32. GetModuleText
33. GetProcText
34. GetProjectText
35. GetSheetByCodeName
36. GreenifyComments
37. GreenifyInlineComments
38. HasProject
39. InStrExact
40. IndentCodeString
41. InitChrWidths
42. IsBlockEnd
43. IsBlockStart
44. IsOnlyLetters
45. LastCell
46. LinkCodeBlocksWithShape
47. ModuleOfProcedure
48. MoveCommentQuoteToActualPosition
49. NumberOfArrayDimensions
50. PrintProject
51. PrintScreen
52. PrinterLastCell
53. PrinterTocAndCode
54. PrinterTocRange
55. ProceduresOfModule
56. ProtectedVBProject
57. PutCodeInPrinter
58. RandomRGB
59. RangeFindAll
60. RangeToString
61. RemoveBlankLines
62. RemoveComments
63. ResetPrinter
64. SetVbeNormal
65. SetVbeOnTop
66. SetXLNormal
67. SetXLOnTop
68. SetupPrinterPage
69. SheetsToPicture
70. ShortenToByteCharacters
71. ShowOnTop
72. SortCollection

73. StartOptimizeCodeRun
74. StartTimer
75. StopOptimizeCodeRun
76. StrWidth
77. TextDecomposition
78. TxtAppend
79. UserformsToPicture
80. WorkbookOfModule
81. WorkbookOfProject
82. closePair
83. dp
84. getLastColumn
85. getLastRow
86. getModuleName
87. getWhichFirstPosition
88. mapOfPageBreaks
89. openPair
90. printArray
91. printCollection
92. printDictionary
93. printRange
94. showPrinter

57 (Module) F_Userforms

1. AddFormControls
2. AddMinimizeButtonToUserform
3. AddMultipleControls
4. CopyControlProperties
5. CreateListboxHeader
6. DeselectListbox
7. DisplayErrorText
8. EditObjectProperties
9. EditObjectsProperty
10. FilterListboxByColumn
11. GetSystemErrorMessageText
12. ListboxContains
13. ListboxSelectedCount
14. ListboxSelectedIndexes
15. ListboxSelectedValues
16. ListboxToRangeSelect
17. LoadPosition
18. LoadUserformOptions
19. MakeFormBorderless
20. MakeFormTransparent
21. MakeUserFormChildOfVBEditor
22. PasteControlProperties
23. Reframe
24. RemoveControlsCaptions
25. RenameControlAndCode
26. ResizeControlColumns
27. ResizeUserformToFitControls
28. SavePosition
29. SaveUserformOptions
30. SelectControItemsByFilter
31. SelectDeselectAll
32. SelectListboxItems
33. SelectedControl
34. SelectedControls
35. SelectedFrameControl
36. SelectedFrameControls

37. SetHandCursor
38. ShowUserform
39. SortControls
40. SortControlsHorizontally
41. SortControlsVertivally
42. SortListboxOnColumn
43. SwitchControlNames
44. SwitchControlPositions
45. TextOfControl
46. TrimToNull
47. UserformOnTop
48. UserformSelectedControlsSetHandCursor
49. UserformSetHandCursor
50. flashControl
51. whichOption

WORKSHEET Snapshots

78 (Image of Worksheet) ProjectManagerTXTColour
79 (Image of Worksheet) ProjectManagerPrinter
80 (Image of Worksheet) README
(Image of Worksheet) Sheet1

USERFORM Snapshots

81 (Image of Userform) uCodePrinter
(Image of Userform) uCodePrinterNavigator
(Image of Userform) uDEV

--- ThisWorkbook ---

--- ProjectManagerTXTColour - Sheet20 ---

--- README - Sheet22 ---

--- Sheet1 - Sheet1 ---

--- ProjectManagerPrinter - Sheet2 ---

--- uCodePrinter ---

```
Private Sub Image1_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, _
ByVal Y As Single)
    Unload Me
End Sub
Private Sub UserForm_Initialize()
    LoadBooksAndAddins listOpenBooks
    SortListboxOnColumn listOpenBooks, 0
    FormatColourFormatters Me
    MakeFormBorderless Me
End Sub
Private Sub SelectFile_Click()
    Dim TargetWorkbook As Workbook
    Dim fPath As String
    fPath = PickExcelFile
    If fPath = "" Then Exit Sub
    Set TargetWorkbook = Workbooks.Open(fileName:=fPath, UpdateLinks:=0, ReadOnly:=False)
    Me.Hide
    PrintProject TargetWorkbook
    Me.Show
End Sub
Private Sub SelectFromList_Click()
    If listOpenBooks.ListIndex = -1 Then
        MsgBox "No book selected"
        Exit Sub
    End If
    Dim TargetWorkbook As Workbook
    Set TargetWorkbook = Workbooks(listOpenBooks.list(listOpenBooks.ListIndex))
    Me.Hide
    PrintProject TargetWorkbook
    Me.Show
End Sub
Private Sub ActiveFile_Click()
    Dim TargetWorkbook As Workbook
    Set TargetWorkbook = ActiveWorkbook
    Me.Hide
    PrintProject TargetWorkbook
    Me.Show
End Sub
Private Sub LBLcolourCode_Click()
    ColorPaletteDialog ThisWorkbook.Sheets("ProjectManagerTXTColour").Range("J1"), LBLcolourCode
End Sub
Private Sub LBLcolourComment_Click()
    ColorPaletteDialog ThisWorkbook.Sheets("ProjectManagerTXTColour").Range("J4"), LBLcolourComment
End Sub
Private Sub LBLcolourKey_Click()
    ColorPaletteDialog ThisWorkbook.Sheets("ProjectManagerTXTColour").Range("J3"), LBLcolourKey
End Sub
Private Sub LBLcolourOdd_Click()
    ColorPaletteDialog ThisWorkbook.Sheets("ProjectManagerTXTColour").Range("J2"), LBLcolourOdd
End Sub
Private Sub cInfo_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, _
ByVal Y As Single)
    uDEV.Show
End Sub
Private Sub LoadBooksAndAddins(TargetControl As MSForms.control)
    Dim coll As New Collection
    Dim wb As Workbook
```

```
    For Each wb In Workbooks
        If Len(wb.Path) > 0 Then
            If ProtectedVBProject(wb) = False Then
                On Error Resume Next
                coll.Add wb.Name, wb.Name
                On Error GoTo 0
            End If
        End If
    Next
    Dim vbProj As VBProject
    Dim wbPath As String
    For Each vbProj In Application.VBE.VBProjects
        On Error GoTo ErrorHandler
        wbPath = vbProj.fileName
        If Right(wbPath, 4) = "xlam" Or Right(wbPath, 3) = "xla" Then
            Dim wbName As String
            wbName = Mid(wbPath, InStrRev(wbPath, "\") + 1)
            If ProtectedVBProject(Workbooks(wbName)) = False Then
                On Error Resume Next
                coll.Add wbName, wbName
                On Error GoTo 0
            End If
        End If
        Skip:
    Next vbProj
    Dim el As Variant
    For Each el In coll
        TargetControl.AddItem el
    Next
    Exit Sub
    ErrorHandler:
    If err.Number = 76 Then GoTo Skip
End Sub
Private Sub SortListboxOnColumn(lBox As MSForms.ListBox, Optional OnColumn As Long = 0)
    Dim vntData As Variant
    Dim vntTempItem As Variant
    Dim lngOuterIndex As Long
    Dim lngInnerIndex As Long
    Dim lngSubItemIndex As Long
    vntData = lBox.list
    For lngOuterIndex = LBound(vntData, 1) To UBound(vntData, 1) - 1
        For lngInnerIndex = lngOuterIndex + 1 To UBound(vntData, 1)
            If vntData(lngOuterIndex, OnColumn) > vntData(lngInnerIndex, OnColumn) Then
                For lngSubItemIndex = 0 To lBox.columnCount - 1
                    vntTempItem = vntData(lngOuterIndex, lngSubItemIndex)
                    vntData(lngOuterIndex, lngSubItemIndex) = vntData(lngInnerIndex,  _
                    lngSubItemIndex)
                    vntData(lngInnerIndex, lngSubItemIndex) = vntTempItem
                Next
            End If
        Next lngInnerIndex
    Next lngOuterIndex
    lBox.clear
    lBox.list = vntData
End Sub
Private Sub FormatColourFormatters(form As Object)
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Sheets("ProjectManagerTXTColour")
    form.LBLcolourCode.ForeColor = ws.Range("J1").Value
```

```
    form.LBLcolourKey.ForeColor = ws.Range("J3").Value
    form.LBLcolourOdd.BackColor = ws.Range("J2").Value
    form.LBLcolourComment.ForeColor = ws.Range("J4").Value
End Sub
Private Sub ColorPaletteDialog(rng As Range, Lbl As MSForms.Label)
    If Application.Dialogs(xlDialogEditColor).Show(10, 0, 125, 125) = True Then
        lcolor = ActiveWorkbook.Colors(10)
        rng.Value = lcolor
        rng.OFFSET(0, 1).Interior.color = lcolor
        Lbl.ForeColor = lcolor
    End If
    ActiveWorkbook.ResetColors
End Sub
```

--- uCodePrinterNavigator ---

```
Private Sub Label1_Click()
    TOCProceduresVisibilityON
End Sub
Private Sub Label2_Click()
    TOCProceduresVisibilityOFF
End Sub
Sub TOCProceduresVisibilityON()
    TOCProceduresVisibility True
End Sub
Sub TOCProceduresVisibilityOFF()
    TOCProceduresVisibility False
End Sub
Sub TOCProceduresVisibility(Display As Boolean)
    Dim rng As Range
    Set rng = PrinterTocRange
    Dim proceduresRange As Range
    Dim cell As Range
    For Each cell In rng
        If Not cell Like "(*" Then
            If proceduresRange Is Nothing Then
                Set proceduresRange = cell
            Else
                Set proceduresRange = Union(proceduresRange, cell)
            End If
        End If
    Next
    proceduresRange.Rows.Hidden = Not Display
End Sub
Private Sub ListBox1_Click()
    changeTOC
End Sub
Private Sub UserForm_Initialize()
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("ProjectManagerPrinter")
    Dim cell As Range
    For Each cell In ws.Columns(2).Cells.SpecialCells(xlCellTypeConstants)
        If cell.Value Like "--- *" Then
            ListBox1.AddItem cell.Value
        End If
    Next
    Me.Top = 100
    Me.Left = 800
    counter = -1
End Sub
Sub changeTOC()
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("ProjectManagerPrinter")
    Dim cell As Range
    Set cell = ws.Columns(2).Find(ListBox1.list(ListBox1.ListIndex))
    cell.Select
    ActiveWindow.ScrollRow = cell.Row
End Sub
```

```
--- uDEV ---
Private Declare PtrSafe Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hWnd As _
Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
Private Declare PtrSafe Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As _
String, ByVal lpWindowName As String) As Long
Private Declare PtrSafe Function FindWindowA Lib "user32" (ByVal lpClassName As String, ByVal _
lpWindowName As String) As Long
Private Declare PtrSafe Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hWnd As _
Long, ByVal nIndex As Long) As Long
Private Declare PtrSafe Function DrawMenuBar Lib "user32" (ByVal hWnd As Long) As Long
Private Declare PtrSafe Function SetLayeredWindowAttributes Lib "user32" (ByVal hWnd As Long, ByVal
crKey As Long, ByVal bAlpha As Byte, ByVal dwFlags As Long) As Long
Private Const GWL_STYLE As Long = (-16)
Private Const GWL_EXSTYLE As Long = (-20)
Private Const WS_CAPTION As Long = &HC00000
Private Const WS_EX_DLGMODALFRAME As Long = &H1
Private Const WS_EX_LAYERED = &H80000
Private Const LWA_COLORKEY = &H1
Private Const LWA_ALPHA = &H2
Private m_sngDownX As Single
Private m_sngDownY As Single
Private Sub MakeFormTransparent(frm As Object, Optional color As Variant)
    Dim formhandle As Long
    Dim bytOpacity As Byte
    formhandle = CLng(FindWindow(vbNullString, frm.Caption))
    If IsMissing(color) Then color = vbWhite
    bytOpacity = 100
    SetWindowLong formhandle, GWL_EXSTYLE, GetWindowLong(formhandle, GWL_EXSTYLE) Or WS_EX_LAYERED
    frm.BackColor = color
    SetLayeredWindowAttributes formhandle, color, bytOpacity, LWA_COLORKEY
End Sub
Private Sub MakeFormBorderless(frm As Object)
    Dim lngWindow As Long
    Dim lFrmHdl As Long
    lFrmHdl = CLng(FindWindow(vbNullString, frm.Caption))
    lngWindow = GetWindowLong(lFrmHdl, GWL_STYLE)
    lngWindow = lngWindow And (Not WS_CAPTION)
    SetWindowLong lFrmHdl, GWL_STYLE, lngWindow
    lngWindow = GetWindowLong(lFrmHdl, GWL_EXSTYLE)
    lngWindow = lngWindow And Not WS_EX_DLGMODALFRAME
    SetWindowLong lFrmHdl, GWL_EXSTYLE, lngWindow
    DrawMenuBar lFrmHdl
End Sub
Private Sub LVK_Click()
    FollowLink ("https://vk.com/vbarc_hive")
End Sub
Private Sub Label2_Click()
    Unload Me
End Sub
Private Sub LFaceBook_Click()
    FollowLink ("https://www.facebook.com/VBA-Code-Archive-110295994460212")
End Sub
Private Sub LGitHub_Click()
    FollowLink ("https://github.com/alexofrhodes")
End Sub
Private Sub LYouTube_Click()
    FollowLink ("https://www.youtube.com/channel/UC5QH3fn1zjx0aUjRER_rOjg")
End Sub
```

```
Private Sub LBuyMeACoffee_Click()
    FollowLink ("http://paypal.me/alexofrhodes")
End Sub
Private Function CLIP(Optional StoreText As String) As String
    Dim X As Variant
    X = StoreText
    With CreateObject("htmlfile")
        With .parentWindow.clipboardData
            Select Case True
            Case Len(StoreText)
                .SetData "text", X
            Case Else
                CLIP = .GetData("text")
            End Select
        End With
    End With
End Function
Private Sub LEmail_Click()
    If GetInternetConnectedState = False Then
        MsgBox "Seems Internet is not available"
        Exit Sub
    End If
    If OutlookCheck = True Then
        MailDev
    Else
        Dim out As String
        out = AUTHOR_EMAIL
        CLIP out
        MsgBox ("Seems Outlook is not available" & Chr(10) & _
        "DEV's email address " & vbNewLine & out & vbNewLine & "copied to clipboard")
    End If
End Sub
Sub MailDev()
    Dim OutApp As Object
    Dim OutMail As Object
    Dim strBody As String
    Set OutApp = CreateObject("Outlook.Application")
    Set OutMail = OutApp.CreateItem(0)
    On Error Resume Next
    With OutMail
        .To = AUTHOR_EMAIL
        .CC = vbNullString
        .BCC = vbNullString
        .Subject = "DEV REQUEST OR FEEDBACK FOR -CODE ARCHIVE-"
        .body = strBody
        .Display
    End With
    On Error GoTo 0
    Set OutMail = Nothing
    Set OutApp = Nothing
End Sub
Private Sub FollowLink(FolderPath As String)
    If Right(FolderPath, 1) = "\" Then FolderPath = Left(FolderPath, Len(FolderPath) - 1)
    On Error Resume Next
    Dim oShell As Object
    Dim Wnd As Object
    Set oShell = CreateObject("Shell.Application")
    For Each Wnd In oShell.Windows
        If Wnd.Name = "File Explorer" Then
```

```
            If Wnd.document.Folder.Self.Path = FolderPath Then Exit Sub
        End If
    Next Wnd
    Application.ThisWorkbook.FollowHyperlink Address:=FolderPath, NewWindow:=True
End Sub
Private Sub UserForm_Initialize()
    MakeFormBorderless Me
    MakeFormTransparent Me, vbBlack
End Sub
```

```
--- Main ---
#If VBA7 Then
    Private Declare PtrSafe Function CloseClipboard Lib "user32" () As Long
    Private Declare PtrSafe Function EmptyClipboard Lib "user32" () As Long
    Private Declare PtrSafe Function OpenClipboard Lib "user32" (ByVal hWnd As Long) As Long
#Else
    Private Declare Function CloseClipboard Lib "user32" () As Long
    Private Declare Function EmptyClipboard Lib "user32" () As Long
    Private Declare Function OpenClipboard Lib "user32" (ByVal hwnd As Long) As Long
#End If
#If Win64 Then
    Private Declare PtrSafe Function SetWindowPos Lib "user32" (ByVal hWnd As LongPtr, ByVal  _
    hWndInsertAfter As LongPtr, ByVal X As Long, ByVal Y As Long, ByVal cx As Long, ByVal cy As  _
    Long, ByVal wFlags As Long) As Long
#Else
    Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long,ByVal hwndInsertAfter As
    Long,ByVal x As Long, ByVal y As Long,ByVal cx As Long, ByVal cy As Long,ByVal wFlags As Long) _
    As Long
#End If
#If VBA7 Then
    Private Declare PtrSafe Sub keybd_event Lib "user32" (ByVal bVk As Byte, ByVal bScan As Byte,  _
    ByVal dwFlags As Long, ByVal dwExtraInfo As LongPtr)
#Else
    Private Declare Sub keybd_event Lib "user32" (ByVal bVk As Byte,  ByVal bScan As Byte, ByVal  _
    dwFlags As Long, ByVal dwExtraInfo As Long)
#End If
Private Const VK_SNAPSHOT = 44
Private Const VK_LMENU = 164
Private Const KEYEVENTF_KEYUP = 2
Private Const KEYEVENTF_EXTENDEDKEY = 1
Public mafChrWid(32 To 127) As Double
Public Const mblncTimer As Boolean = True
Public mvarTimerName
Public mvarTimerStart
Sub showPrinter()
    uCodePrinter.Show
End Sub
Public Sub PrintProject(WorkbookOrPath As Variant)
    Dim TargetWorkbook As Workbook
    Dim wasOpen As Boolean
    On Error Resume Next
    Set TargetWorkbook = WorkbookOrPath
    If Not TargetWorkbook Is Nothing Then
        wasOpen = True
    Else
        wasOpen = False
        Set TargetWorkbook = Workbooks.Open(WorkbookOrPath)
        If Not TargetWorkbook Is Nothing Then wasOpen = True
        If TargetWorkbook Is Nothing Then Set TargetWorkbook = Workbooks.Open(WorkbookOrPath)
    End If
    If TargetWorkbook Is Nothing Then
        MsgBox "No valid workbook or path passed."
        Exit Sub
    End If
    On Error GoTo 0
    If ProtectedVBProject(TargetWorkbook) = True Or HasProject(TargetWorkbook) = False Then
        MsgBox "Project Empty or Protected"
        Exit Sub
```

```
    End If
    Dim wasAddin As Boolean
    wasAddin = TargetWorkbook.IsAddin
    TargetWorkbook.IsAddin = False
    StartTimer "PrintProject"
    ResetPrinter
    PutCodeInPrinter PrinterTocAndCode(TargetWorkbook), True
    ApplyPrinterTableStyle
    AddLogoToFirstPage TargetWorkbook
    FormatTextColors
    FormatPrinterTitles
    LinkCodeBlocksWithShape
    SheetsToPicture TargetWorkbook, PrinterLastCell.OFFSET(1)
    UserformsToPicture TargetWorkbook, PrinterLastCell.OFFSET(1)
    AddPageBreaksToPrinter
    SetupPrinterPage TargetWorkbook
    AssignPageNumbersToToc
    ThisWorkbook.Sheets("ProjectManagerPrinter").Copy
    EndTimer
    SetXLNormal
    If Not wasOpen Then
        TargetWorkbook.Close False
    Else
        TargetWorkbook.IsAddin = wasAddin
    End If
End Sub
Private Function StartTimer(TimerName)
    On Error GoTo ERR_HANDLER
    If mblncTimer Then
        mvarTimerName = TimerName
        mvarTimerStart = Timer
    End If
    On Error Resume Next
    Exit Function
    ERR_HANDLER:
    MsgBox err.Number & " " & err.Description, vbCritical, "StartTimer()"
End Function
Private Function EndTimer()
    On Error GoTo ERR_HANDLER
    Dim strFile As String
    Dim strContent As String
    If mblncTimer Then
        Dim strPath As String
        strPath = Environ("USERprofile") & "\My Documents\vbArc\Timers\"
        FoldersCreate strPath
        strFile = strPath & mvarTimerName & ".txt"
        & Left(ThisWorkbook.Name, InStr(1, ThisWorkbook.Name, ".") - 1) _
        & "TimerLog.txt"
        If Len(Dir(strFile)) = 0 Then
            strContent = _
            "Timestamp" & vbTab & vbTab & vbTab & vbTab & _
            "ElapsedTime" & vbTab & vbTab & _
            "TimerName"
            TxtAppend strFile, strContent
        End If
        strContent = Now() & vbTab & vbTab & _
        Format(Timer - mvarTimerStart, "0.00") & vbTab & vbTab & vbTab & _
        mvarTimerName
        TxtAppend strFile, strContent
```

```
    End If
    On Error Resume Next
    Exit Function
    ERR_HANDLER:
    MsgBox err.Number & " " & err.Description, vbCritical, "EndTimer()"
End Function
Private Sub StartOptimizeCodeRun()
    Application.ScreenUpdating = False
    Application.DisplayStatusBar = False
    Application.Calculation = xlCalculationManual
    Application.EnableEvents = False
    ActiveSheet.DisplayPageBreaks = False
End Sub
Private Sub StopOptimizeCodeRun()
    Application.ScreenUpdating = True
    Application.DisplayStatusBar = True
    Application.Calculation = xlCalculationAutomatic
    Application.EnableEvents = True
    ActiveSheet.DisplayPageBreaks = False
End Sub
Private Function HasProject(wb As Workbook) As Boolean
    Dim WbProjComp As Object
    On Error Resume Next
    Set WbProjComp = wb.VBProject.VBComponents
    If Not WbProjComp Is Nothing Then HasProject = True
End Function
Public Function ProtectedVBProject(ByVal wb As Workbook) As Boolean
    If wb.VBProject.Protection = 1 Then
        ProtectedVBProject = True
    Else
        ProtectedVBProject = False
    End If
End Function
Private Function PrinterLastCell() As Range
    Dim TargetWorksheet As Worksheet
    Set TargetWorksheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    Set PrinterLastCell = TargetWorksheet.Range("B" & getLastRow(TargetWorksheet))
End Function
Private Function PrinterTocAndCode(TargetWorkbook As Workbook, _
    Optional includeCode As Boolean = True, _
    Optional includeTOC As Boolean = True, _
    Optional includeTocProcedures As Boolean = True)
    Dim Module As VBComponent
    Dim ModuleTypes
    ModuleTypes = Array(vbext_ct_Document, vbext_ct_MSForm, vbext_ct_StdModule,  _
    vbext_ct_ClassModule)
    Dim ModuleType
    Dim TargetWorksheet As Worksheet
    Dim TargetWorkSheetName As String
    Dim Contents As Collection
    Set Contents = New Collection
    Dim Procedure As Variant
    Dim code As Variant
    Dim i As Long
    Dim counter As Long
    If includeTOC Then
        Contents.Add "--- Table of Contents: ---"
        For Each ModuleType In ModuleTypes
            For Each Module In TargetWorkbook.VBProject.VBComponents
```

```
            If Module.Type = ModuleType Then
                If ModuleType = vbext_ct_Document And Module.Name <> "ThisWorkbook" Then
                    TargetWorkSheetName = GetSheetByCodeName(TargetWorkbook, Module.Name).Name
                    Contents.Add "(" & ComponentTypeToString(Module.Type) & ")" & " " &  _
                    TargetWorkSheetName & " - " & Module.Name
                Else
                    Contents.Add "(" & ComponentTypeToString(Module.Type) & ")" & " " & Module. _
                    Name
                End If
                If Module.CodeModule.CountOfLines > 0 Then
                    If includeTocProcedures Then
                        counter = 0
                        For Each Procedure In SortCollection(ProceduresOfModule(Module))
                            counter = counter + 1
                            Contents.Add Space(4) & counter & ". " & Procedure
                        Next
                    End If
                End If
            End If
        Next Module
    Next ModuleType
    Contents.Add ""
    Contents.Add "WORKSHEET Snapshots"
    Contents.Add ""
    For Each TargetWorksheet In TargetWorkbook.Worksheets
        Contents.Add "(Image of Worksheet) " & TargetWorksheet.Name
    Next
    Contents.Add ""
    Contents.Add "USERFORM Snapshots"
    Contents.Add ""
    For Each Module In TargetWorkbook.VBProject.VBComponents
        If Module.Type = vbext_ct_MSForm Then
            Contents.Add "(Image of Userform) " & Module.Name
        End If
    Next
End If
Dim output As String
If includeCode Then
    For Each ModuleType In ModuleTypes
        For Each Module In TargetWorkbook.VBProject.VBComponents
            If Module.Type = ModuleType Then
                If ModuleType = vbext_ct_Document And Module.Name <> "ThisWorkbook" Then
                    TargetWorkSheetName = GetSheetByCodeName(TargetWorkbook, Module.Name).Name
                    Contents.Add "--- " & TargetWorkSheetName & " - " & Module.Name & " ---"
                Else
                    Contents.Add "--- " & Module.Name & " ---"
                End If
                If Module.CodeModule.CountOfLines > 0 Then
                    code = GetCodeOf(Module)
                    code = IndentCodeString(code)
                    code = R
                    code = RemoveBlankLines(code)
                    code = Split(code, vbNewLine)
                    For i = LBound(code) To UBound(code)
                        Contents.Add code(i)
                    Next i
                End If
            End If
        Next Module
```

```
        Next ModuleType
        PrinterTocAndCode = CollectionToArray(Contents)
    End If
End Function
Private Sub AssignPageNumbersToToc()
    Dim TargetWorksheet As Worksheet
    Set TargetWorksheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    Dim dic As New Dictionary
    Set dic = mapOfPageBreaks(TargetWorksheet)
    Dim FindWhat
    Dim cell As Range
    For Each cell In PrinterTocRange
        If Left(cell.TEXT, 1) = "(" Then
            If InStr(1, cell.TEXT, "Image of Worksheet") > 0 Then
                FindWhat = "--- Image of Worksheet : "
            ElseIf InStr(1, cell.TEXT, "Image of Userform") > 0 Then
                FindWhat = "--- Image of Userform : "
            Else
                FindWhat = "--- "
            End If
            FindWhat = FindWhat & Split(cell.TEXT, ") ")(1) & " ---"
            cell.OFFSET(0, -1).Value = dic(FindWhat)
        End If
    Next
End Sub
Private Sub PutCodeInPrinter(TextToPrinter As Variant, singleColumn As Boolean)
    If TypeName(TextToPrinter) = "String" Then TextToPrinter = Split(TextToPrinter, vbNewLine)
    Dim var
    var = TextToPrinter
    Dim TargetSheet As Worksheet
    Set TargetSheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    ArrayToRange2D var, TargetSheet.Cells(2, 2)
    TargetSheet.Cells.WrapText = False
    BreakText
    If Not singleColumn Then
        var = TargetSheet.Range("B2:B" & getLastRow(TargetSheet))
        var = WorksheetFunction.Transpose(var)
        var = Array1dTo2dByIndentation(TextToPrinter)
        TargetSheet.Cells.clear
        ArrayToRange2D var, TargetSheet.Cells(2, 2)
        TargetSheet.Cells.EntireColumn.ColumnWidth = 4.5
    End If
    If WorksheetFunction.CountA(TargetSheet.Columns(3)) = 0 Then TargetSheet.Columns.AutoFit
End Sub
Private Sub LinkCodeBlocksWithShape()
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Sheets("ProjectManagerPrinter")
    Dim singleColumn As Boolean
    singleColumn = (WorksheetFunction.CountA(ws.Columns(3)) = 0)
    Dim ShapeTypeNumber As Long
    ShapeTypeNumber = 29
    Dim CloseTXT As String
    Dim X As Variant
    Dim shp As Shape
    Dim trimCell As String
    Dim counter As Long
    Dim cell As Range
    Dim colNo As Long
    For colNo = 2 To getLastColumn(ws)
```

```
        For Each cell In ws.Columns(colNo).SpecialCells(xlCellTypeConstants)
            trimCell = Trim(cell.TEXT)
            If IsBlockStart(trimCell) Then
                Select Case openPair(trimCell)
                Case Is = "Case"
                    GoTo Skip
                Case Is = "Else"
                Case Is = "If"
                    If Right(trimCell, 4) <> "Then" Then GoTo Skip
                Case Is = "skip"
                    GoTo Skip
                Case Else
                End Select
                CloseTXT = closePair(trimCell)
                counter = Len(cell) - Len(trimCell)
                On Error Resume Next
                Dim foundMatch As Range
                Set foundMatch = ws.Columns(colNo).Find( _
                IIf(singleColumn = True, Space(counter), "") & _
                CloseTXT & "*", _
                After:=cell, _
                LookAt:=xlWhole)
                On Error GoTo 0
                If foundMatch Is Nothing Then GoTo Skip
                If foundMatch.Row > cell.Row Then
                    Set shp = ws.Shapes.AddShape( _
                    ShapeTypeNumber, _
                    cell.Left - 5, _
                    cell.Top + (cell.Height / 2), _
                    5, _
                    ws.Range(cell, foundMatch).Height - cell.Height)
                    If singleColumn Then
                        X = StrWidth(Application.WorksheetFunction.Rept("A", counter), "Consolas",  _
                        10)
                        shp.Left = shp.Left + X
                    End If
                End If
            End If
            Skip:
        Next cell
    Next colNo
    ColorizeBlockLinksByLevel
End Sub
Private Sub AddPageBreaksToPrinter()
    Dim TargetSheet As Worksheet
    Set TargetSheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    TargetSheet.ResetAllPageBreaks
    Dim rng As Range
    Set rng = FindAll(TargetSheet.Columns(2).SpecialCells(xlCellTypeConstants), "---", , xlPart, , ,
    "--- ")
    Dim cell As Range
    For Each cell In rng
        TargetSheet.Rows(cell.Row).PageBreak = xlPageBreakManual
    Next
End Sub
Private Sub ApplyPrinterTableStyle()
    On Error Resume Next
    ThisWorkbook.TableStyles("vbArcPrinterStyle").Delete
    On Error GoTo 0
```

```
    Dim TargetSheet As Worksheet
    Set TargetSheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    Dim TargetRange As Range
    Set TargetRange = TargetSheet.Range("A2:" & TargetSheet.Cells(getLastRow(TargetSheet),  _
    getLastColumn(TargetSheet)).Address)
    Dim myFirstRowColor As Long
    myFirstRowColor = ThisWorkbook.Sheets("ProjectManagerTXTColour").Range("J2").Value
    Dim TS As TableStyle
    Set TS = ThisWorkbook.TableStyles.Add("vbArcPrinterStyle")
    TS.TableStyleElements(xlRowStripe1).Interior.color = myFirstRowColor
    TS.ShowAsAvailableTableStyle = True
    Dim TB As ListObject
    Set TB = TargetSheet.ListObjects.Add(xlSrcRange, TargetRange, , xlNo, , TS.Name)
    TB.Unlist
    TargetSheet.Rows(2).Delete
    TS.Delete
    TargetSheet.Rows(2).ClearFormats
    TargetSheet.Columns(1).ColumnWidth = 4
    PrinterTocRange.EntireRow.Interior.ColorIndex = 0
End Sub
Private Sub FormatPrinterTitles()
    Dim cell As Range
    Dim TargetSheet As Worksheet
    Set TargetSheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    Dim rng As Range
    For Each cell In FindAll(TargetSheet.Columns(2).SpecialCells(xlCellTypeConstants), "---", ,  _
    xlPart)
        If Left(Trim(cell.Value), 3) = "---" Then
            If rng Is Nothing Then
                Set rng = cell
            Else
                Set rng = Union(rng, cell)
            End If
        End If
    Next
    rng.Font.Size = 18
    rng.Font.Bold = True
    rng.Font.color = vbBlack
End Sub
Private Sub AddLogoToFirstPage(TargetWorkbook As Workbook, Optional LogoPath As String)
    ClearClipboard
    Dim PrinterSheet As Worksheet
    Set PrinterSheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    PrinterSheet.Rows(1).EntireRow.Insert
    PrinterSheet.Rows(1).Interior.ColorIndex = 0
    Dim targetCell As Range
    Set targetCell = PrinterSheet.Range("B1")
    Dim shp As Shape
    If LogoPath = "" Then
        ImageFailLoad:
        ThisWorkbook.Sheets("README").Shapes("LOGO").Copy
        PrinterSheet.Paste targetCell
        Set shp = PrinterSheet.Shapes("LOGO")
    Else
        On Error Resume Next
        PrinterSheet.Pictures.Insert LogoPath
        On Error GoTo 0
        If err.Number <> 0 Then GoTo ImageFailLoad
        Set shp = PrinterSheet.Shapes(1)
```

```
    End If
    With targetCell
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlVAlignBottom
        .WrapText = False
        .Value = "vbArc > Code Printer > " & TargetWorkbook.Name
        .Characters.Font.Size = 12
        .Characters.Font.Bold = True
        .Characters.Font.ColorIndex = 10
        .Characters.Font.Name = "Comic Sans MS"
        .RowHeight = 330
    End With
    shp.LockAspectRatio = True
    shp.Name = "LOGO"
    shp.Top = targetCell.Top
    shp.Height = targetCell.EntireRow.Height - 20
    shp.Left = targetCell.Width / 2 - shp.Width / 2
End Sub
Sub SetupPrinterPage(TargetWorkbook As Workbook)
    Dim PrinterSheet As Worksheet
    Set PrinterSheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    Dim fileName As String
    fileName = TargetWorkbook.Name
    With PrinterSheet.PageSetup
        .PrintArea = PrinterSheet.Range("A1:" & Cells(getLastRow(PrinterSheet), getLastColumn( _
        PrinterSheet)).Address).Address
        .LeftMargin = Application.InchesToPoints(0.25)
        .RightMargin = Application.InchesToPoints(0.25)
        .TopMargin = Application.InchesToPoints(0.25)
        .BottomMargin = Application.InchesToPoints(0.75)
        .LeftFooter = fileName
        .CenterFooter = "Page &P of &N"
        .RightFooter = "&D"
        .FitToPagesWide = 1
        .FitToPagesTall = False
    End With
End Sub
Private Sub FormatTextColors()
    Dim PrinterSheet As Worksheet
    Set PrinterSheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    Dim ColorSheet As Worksheet
    Set ColorSheet = ThisWorkbook.Sheets("ProjectManagerTXTColour")
    Dim myColor As Long
    myColor = ColorSheet.Range("J1").Value
    PrinterSheet.Cells.Font.color = myColor
    myColor = ColorSheet.Range("J3").Value
    Dim MyWords As Variant
    MyWords = Split(RangeToString(ColorSheet.Range("A1").CurrentRegion), ",")
    Dim printRange As Range
    Set printRange = PrinterSheet.UsedRange.SpecialCells(xlCellTypeConstants)
    ColorWords printRange, MyWords, myColor
End Sub
Private Sub ResetPrinter()
    Dim PrinterSheetName As String
    PrinterSheetName = "ProjectManagerPrinter"
    DeleteWorksheet ThisWorkbook.Sheets(PrinterSheetName)
    ThisWorkbook.Sheets.Add().Name = PrinterSheetName
    With ThisWorkbook.Sheets(PrinterSheetName)
        .Cells.VerticalAlignment = xlVAlignTop
```

```
        .Cells.Font.Name = "Consolas"
        .Columns(1).ColumnWidth = 4
    End With
End Sub
Private Sub UserformsToPicture(TargetWorkbook As Workbook, TargetRange As Range)
    Dim PrinterSheet As Worksheet
    Set PrinterSheet = TargetRange.parent
    Dim myPicture As Shape
    Dim Module As VBComponent
    SetVbeOnTop
    For Each Module In TargetWorkbook.VBProject.VBComponents
        If Module.Type = vbext_ct_MSForm Then
            ClearClipboard
            TargetRange.Value = "--- Image of Userform : " & Module.Name & " ---"
            Set TargetRange = TargetRange.OFFSET(2)
            Module.Activate
            DoEvents
            Application.Wait (Now + TimeValue("0:00:2"))
            PrintScreen
            Application.Wait (Now + TimeValue("0:00:2"))
            ThisWorkbook.VBProject.VBComponents("U_ProjectManager").Activate
            DoEvents
            Application.Wait (Now + TimeValue("0:00:1"))
            TargetRange.PasteSpecial
            Set myPicture = PrinterSheet.Shapes(PrinterSheet.Shapes.count)
            myPicture.Name = "Image of Worksheet " & Module.Name
            myPicture.Top = TargetRange.Top
            myPicture.Left = TargetRange.Left + 10
            myPicture.Width = PrinterSheet.Range("B1:B" & getLastColumn(PrinterSheet)).Width - 20
            Set TargetRange = TargetRange.OFFSET(myPicture.BottomRightCell.Row - myPicture. _
            TopLeftCell.Row + 1)
            TargetRange.Value = "PictureEnd"
            Set TargetRange = TargetRange.OFFSET(2)
        End If
    Next
    SetVbeNormal
End Sub
Private Sub SheetsToPicture(TargetWorkbook As Workbook, TargetRange As Range)
    Dim PrinterSheet As Worksheet
    Set PrinterSheet = TargetRange.parent
    Dim myPicture As Shape
    Dim ws As Worksheet
    SetXLOnTop
    For Each ws In TargetWorkbook.Worksheets
        If ws.visible Then
            ClearClipboard
            TargetRange.Value = "--- Image of Worksheet : " & ws.Name & " ---"
            Set TargetRange = TargetRange.OFFSET(2)
            ws.Activate
            DoEvents
            Application.Wait (Now + TimeValue("0:00:2"))
            ActiveWindow.WindowState = xlMaximized
            ActiveWindow.ScrollRow = 1
            ActiveWindow.ScrollColumn = 1
            ActiveWindow.Zoom = 100
            PrintScreen
            Application.Wait (Now + TimeValue("0:00:2"))
            PrinterSheet.Range(TargetRange.Address).PasteSpecial
            Set myPicture = PrinterSheet.Shapes(PrinterSheet.Shapes.count)
```

```
            myPicture.LockAspectRatio = True
            myPicture.Name = "Image of Worksheet " & ws.Name
            myPicture.Top = TargetRange.Top
            myPicture.Width = PrinterSheet.Range("B1:B" & getLastColumn(PrinterSheet)).Width - 20
            myPicture.Left = TargetRange.Left + 10
            Set TargetRange = PrinterSheet.Cells(myPicture.BottomRightCell.Row, 2)
            TargetRange.Value = "PictureEnd"
            Set TargetRange = TargetRange.OFFSET(2)
        End If
    Next
    SetXLNormal
End Sub
Private Sub SetXLNormal()
    ShowOnTop Application.hWnd, False
End Sub
Private Function getLastRow(TargetSheet As Worksheet)
    Dim LastCell As Range
    On Error Resume Next
    Set LastCell = TargetSheet.Cells.Find("*", SearchOrder:=xlByRows, SearchDirection:=xlPrevious)
    On Error GoTo 0
    If LastCell Is Nothing Then
        getLastRow = 1
    Else
        getLastRow = LastCell.Row
    End If
End Function
Private Function IndentCodeString(str As Variant)
    If TypeName(str) Like "String*" Then str = Split(str, vbNewLine)
    Dim strNewLine As String
    Dim nIndent As Integer
    Dim i As Long
    For i = LBound(str) To UBound(str)
        strNewLine = str(i)
        strNewLine = LTrim$(strNewLine)
        If IsBlockEnd(strNewLine) Then
            nIndent = nIndent - 1
        End If
        If nIndent < 0 Then
            nIndent = 0
        End If
        If strNewLine <> "" Then
            str(i) = Space$(nIndent * 4) & strNewLine
        End If
        If IsBlockStart(LTrim$(strNewLine)) Then
            nIndent = nIndent + 1
        End If
    Next
    IndentCodeString = Join(str, vbNewLine)
End Function
Private Function ProceduresOfModule(Module As VBComponent) As Collection
    Dim ProcKind As VBIDE.vbext_ProcKind
    Dim LineNum As Long
    Dim coll As New Collection
    Dim procName As String
    With Module.CodeModule
        LineNum = .CountOfDeclarationLines + 1
        Do Until LineNum >= .CountOfLines
            procName = .ProcOfLine(LineNum, ProcKind)
            coll.Add procName
```

```
            LineNum = .ProcStartLine(procName, ProcKind) + .ProcCountLines(procName, ProcKind) + 1
        Loop
    End With
    Set ProceduresOfModule = coll
End Function
Private Function CommentsToOwnLine(str As Variant) As String
    Dim var As Variant
    ReDim var(0)
    If TypeName(str) Like "String*" Then str = Split(str, vbNewLine)
    Dim n               As Long
    Dim i               As Long
    Dim j               As Long
    Dim k               As Long
    Dim l               As Long
    Dim LineText        As String
    Dim QUOTES          As Long
    Dim Q               As Long
    Dim StartPos        As Long
    For j = LBound(str) To UBound(str)
        LineText = Trim(str(j))
        StartPos = 1
        RETRY:
        n = InStr(StartPos, LineText, "'")
        Q = InStr(StartPos, LineText, """")
        QUOTES = 0
        If Q < n Then
            For l = 1 To n
                If Mid(LineText, l, 1) = """" Then
                    QUOTES = QUOTES + 1
                End If
            Next l
        End If
        If QUOTES = Application.WorksheetFunction.Odd(QUOTES) Then
            StartPos = n + 1
            GoTo RETRY:
        Else
            Select Case n
            Case Is = 0
                var(UBound(var)) = str(j)
                ReDim Preserve var(UBound(var) + 1)
            Case Is = 1
                var(UBound(var)) = MoveCommentQuoteToActualPosition(Array(str(j)))
                ReDim Preserve var(UBound(var) + 1)
            Case Is > 1
                var(UBound(var)) = Space(Len(str(j)) - Len(LTrim(str(j)))) & Mid(LineText, n)
                ReDim Preserve var(UBound(var) + 1)
                var(UBound(var)) = Space(Len(str(j)) - Len(LTrim(str(j)))) & Left(LineText, n - 1)
                ReDim Preserve var(UBound(var) + 1)
            End Select
        End If
    Next j
    CommentsToOwnLine = Join(var, vbNewLine)
    CommentsToOwnLine = Left(CommentsToOwnLine, Len(CommentsToOwnLine) - Len(vbNewLine))
End Function
Private Function RemoveComments(str As Variant) As String
    Dim var As Variant
    ReDim var(0)
    If TypeName(str) Like "String*" Then
        str = Split(str, vbNewLine)
```

```
    End If
    Dim n               As Long
    Dim i               As Long
    Dim j               As Long
    Dim k               As Long
    Dim l               As Long
    Dim LineText        As String
    Dim QUOTES          As Long
    Dim Q               As Long
    Dim StartPos        As Long
    For j = LBound(str) To UBound(str)
        LineText = Trim(str(j))
        If LineText Like "Rem *" Then GoTo Skip
        StartPos = 1
        RETRY:
        n = InStr(StartPos, LineText, "'")
        Q = InStr(StartPos, LineText, """")
        QUOTES = 0
        If Q < n Then
            For l = 1 To n
                If Mid(LineText, l, 1) = """" Then
                    QUOTES = QUOTES + 1
                End If
            Next l
        End If
        If QUOTES = Application.WorksheetFunction.Odd(QUOTES) Then
            StartPos = n + 1
            GoTo RETRY:
        Else
            Select Case n
            Case Is = 0
                If Len(LineText) > 0 Then
                    var(UBound(var)) = str(j)
                    ReDim Preserve var(UBound(var) + 1)
                End If
            Case Is = 1
            Case Is > 1
                var(UBound(var)) = Left(str(j), n - 1)
                ReDim Preserve var(UBound(var) + 1)
            End Select
        End If
        Skip:
    Next j
    RemoveComments = Join(var, vbNewLine)
    If RemoveComments = "" Then Exit Function
    RemoveComments = Left(RemoveComments, Len(RemoveComments) - Len(vbNewLine))
End Function
Private Function RemoveBlankLines(str As Variant)
    Dim var As Variant
    ReDim var(0)
    If TypeName(str) Like "String*" Then str = Split(str, vbNewLine)
    For j = LBound(str) To UBound(str)
        LineText = Trim(str(j))
        If Len(Trim(LineText)) > 0 Then
            var(UBound(var)) = str(j)
            ReDim Preserve var(UBound(var) + 1)
        End If
    Next
    RemoveBlankLines = Join(var, vbNewLine)
```

```
    If RemoveBlankLines = "" Then Exit Function
    RemoveBlankLines = Left(RemoveBlankLines, Len(RemoveBlankLines) - Len(vbNewLine))
End Function
Private Function GetCodeOf(ProcedureModuleWorkbook As Variant, Optional WorkbookOfProcedure As _
Workbook, Optional ComponentOfProcedure As VBComponent) As String
    GetCodeOf = ""
    If WorkbookOfProcedure Is Nothing Then Set WorkbookOfProcedure = ActiveCodepaneWorkbook
    On Error GoTo ErrorHandler
    Dim ErrorLocation As String
    Select Case TypeName(ProcedureModuleWorkbook)
    Case "Workbook"
        ErrorLocation = "Workbook"
        GetCodeOf = GetProjectText(ProcedureModuleWorkbook)
    Case "VBComponent"
        ErrorLocation = "Module"
        GetCodeOf = GetCompText(ProcedureModuleWorkbook)
    Case "String"
        ErrorLocation = "Procedure"
        If ComponentOfProcedure Is Nothing Then Set ComponentOfProcedure = ModuleOfProcedure( _
        WorkbookOfProcedure, ProcedureModuleWorkbook)
        GetCodeOf = GetProcText(ComponentOfProcedure, ProcedureModuleWorkbook)
    End Select
    Exit Function
    ErrorHandler:
    Dim errorMessage As String
    Select Case ErrorLocation
    Case Is = "Workbook"
        errorMessage = "Project not found or protected in : " & ProcedureModuleWorkbook.Name
    Case "Module"
        errorMessage = "Module not found : " & ProcedureModuleWorkbook.Name
    Case "Procedure"
        errorMessage = "Procedure not found " & ProcedureModuleWorkbook
    End Select
    Debug.Print errorMessage
    Resume Next
End Function
Private Function ComponentTypeToString(componentType As VBIDE.vbext_ComponentType) As String
    Select Case componentType
    Case vbext_ct_ActiveXDesigner
        ComponentTypeToString = "ActiveX Designer"
    Case vbext_ct_ClassModule
        ComponentTypeToString = "Class"
    Case vbext_ct_Document
        ComponentTypeToString = "Document"
    Case vbext_ct_MSForm
        ComponentTypeToString = "UserForm"
    Case vbext_ct_StdModule
        ComponentTypeToString = "Module"
    Case Else
        ComponentTypeToString = "Unknown Type: " & CStr(componentType)
    End Select
End Function
Private Function GetSheetByCodeName(wb As Workbook, CodeName As String) As Worksheet
    Dim sh As Worksheet
    For Each sh In wb.Worksheets
        If UCase(sh.CodeName) = UCase(CodeName) Then Set GetSheetByCodeName = sh: Exit For
    Next sh
End Function
Private Sub BreakText()
```

```
    Dim cell      As Range
    Dim tmpString    As String
    Dim Splitter  As Long
    Dim counter   As Long
    Dim limit     As Long
    Dim splitOnArray As Variant
    splitOnArray = Array("(", ")", ".", ",", "=", " ")
    If ThisWorkbook.Sheets("ProjectManagerPrinter").PageSetup.Orientation = xlPortrait Then
        limit = 100
    Else
        limit = 140
    End If
    Dim rng As Range
    With ThisWorkbook.Sheets("ProjectManagerPrinter")
        Set rng = .Range("B1:B" & .Range("B" & .Rows.count).End(xlUp).Row)
    End With
    Dim coll As Collection
    Set coll = New Collection
    On Error Resume Next
    For Each cell In rng
        tmpString = cell.TEXT
        REPEATME:
        If Len(tmpString) > limit Then
            counter = Len(tmpString) - Len(Trim(tmpString))
            Splitter = getWhichFirstPosition(tmpString, splitOnArray, limit, False)
            coll.Add Left(tmpString, Splitter) & " _"
            tmpString = Space(counter) & Trim(Mid(tmpString, Splitter + 1))
            GoTo REPEATME
        Else
            coll.Add (tmpString)
        End If
        Skip:
    Next cell
    Dim Arr
    Arr = CollectionToArray(coll)
    With ThisWorkbook.Sheets("ProjectManagerPrinter")
        .Cells.clear
        .Range("B1:B" & UBound(Arr) + 1).Value = WorksheetFunction.Transpose(Arr)
        .Cells.Font.Name = "Consolas"
    End With
End Sub
Private Function SortCollection(colInput As Collection) As Collection
    Dim iCounter As Integer
    Dim iCounter2 As Integer
    Dim Temp As Variant
    Set SortCollection = New Collection
    For iCounter = 1 To colInput.count - 1
        For iCounter2 = iCounter + 1 To colInput.count
            If colInput(iCounter) > colInput(iCounter2) Then
                Temp = colInput(iCounter2)
                colInput.Remove iCounter2
                colInput.Add Temp, , iCounter
            End If
        Next iCounter2
    Next iCounter
    Set SortCollection = colInput
End Function
Private Function CollectionToArray(c As Collection) As Variant
    Dim a() As Variant: ReDim a(0 To c.count - 1)
```

```
    Dim i As Long
    For i = 1 To c.count
        a(i - 1) = c.Item(i)
    Next
    CollectionToArray = a
End Function
Private Function mapOfPageBreaks(TargetSheet As Worksheet)
    Dim dic As New Dictionary
    Dim rng As Range
    Dim prntArea As Range
    Set prntArea = TargetSheet.Range(TargetSheet.PageSetup.PrintArea)
    If TargetSheet.HPageBreaks.count = 0 T
    Set rng = prntArea
End If
If TargetSheet.HPageBreaks.count = 1 Then
Set rng = prntArea.Cells(1, 1)
Set rng = TargetSheet.Range(rng.Address, TargetSheet.HPageBreaks(1).Location.OFFSET(-1).Address)
Set rng = Application.Intersect(rng.EntireRow, prntArea)
Set rng = TargetSheet.Range(TargetSheet.HPageBreaks(1).Location.Address, prntArea.Cells(prntArea. _
Rows.count, 1))
Set rng = Application.Intersect(rng.EntireRow, prntArea)
End If
If TargetSheet.HPageBreaks.count > 1 Then
Set rng = prntArea.Cells(1, 1)
Set rng = TargetSheet.Range(rng.Address, TargetSheet.HPageBreaks(1).Location.OFFSET(-1).Address)
Set rng = TargetSheet.Range(rng.Address, TargetSheet.Cells(prntArea.Cells(1, 1).Row, prntArea.Cells(
prntArea.count).Column).Address)
Dim pgBreaks As Integer
For pgBreaks = 1 To TargetSheet.HPageBreaks.count - 1
    Set rng = TargetSheet.Range(TargetSheet.HPageBreaks(pgBreaks).Location.Address, TargetSheet. _
    HPageBreaks(pgBreaks + 1).Location.OFFSET(-1).Address)
    If rng.Columns.count < prntArea.Columns.count Then
        Set rng = rng.RESIZE(rng.Rows.count, prntArea.Columns.count)
    End If
    If rng.Cells(1, 2).Value Like "--- *" Then dic.Add rng.Cells(1, 2).Value, pgBreaks
Next pgBreaks
Set rng = TargetSheet.Range(TargetSheet.HPageBreaks(TargetSheet.HPageBreaks.count).Location.Address,
prntArea.Cells(prntArea.Rows.count, 1))
Set rng = Application.Intersect(rng.EntireRow, prntArea)
If rng.Cells(1, 2).Value Like "--- *" Then dic.Add rng.Cells(1, 2).Value, pgBreaks
End If
Set mapOfPageBreaks = dic
End Function
Private Function PrinterTocRange() As Range
    Dim TargetWorksheet As Worksheet
    Set TargetWorksheet = ThisWorkbook.Sheets("ProjectManagerPrinter")
    Dim rng As Range
    Set rng = TargetWorksheet.Columns(2).Find("--- Table of Contents", LookAt:=xlPart).OFFSET(1, 0)
    Dim untilCell As Range
    Set untilCell = TargetWorksheet.Columns(2).Find("---", After:=rng, LookAt:=xlPart).OFFSET(-1, 0)
    Set PrinterTocRange = Range(rng, untilCell)
End Function
Private Function Array1dTo2dByIndentation(sourceCode As Variant) As Variant
    Dim var()
    ReDim var(0 To UBound(sourceCode), 0 To 0)
    Dim i As Long
    Dim off As Long
    For i = LBound(sourceCode) To UBound(sourceCode)
        off = (Len(sourceCode(i)) - Len(LTrim(sourceCode(i)))) / 4
```

```
        If off > UBound(var, 2) Then
            ReDim Preserve var(0 To UBound(sourceCode), 0 To off)
        End If
        var(i, IIf(off = 0, 0, off)) = LTrim(sourceCode(i))
    Next
    Array1dTo2dByIndentation = var
End Function
Private Sub ArrayToRange2D(arr2d As Variant, rng As Range)
    If ArrayDimensionLength(arr2d) = 1 Then arr2d = WorksheetFunction.Transpose(arr2d)
    Dim dif As Long
    dif = IIf(LBound(arr2d, 1) = 0, 1, 0)
    rng.RESIZE(UBound(arr2d, 1) + dif, UBound(arr2d, 2) + dif).Value = arr2d
End Sub
Private Function IsBlockStart(strLine As String) As Boolean
    strLine = Replace(strLine, Chr(13), "")
    Dim bOK As Boolean
    Dim nPos As Integer
    Dim strTemp As String
    nPos = InStr(1, strLine, " ") - 1
    If nPos < 0 Then nPos = Len(strLine)
    strTemp = Left$(strLine, nPos)
    Select Case strTemp
    Case "With", "For", "Do", "While", "Select", "Case", "Else", "Else:", "#Else", "#Else:", "Sub",
    "Function", "Property", "Enum", "Type"
        bOK = True
    Case "If", "#If", "ElseIf", "#ElseIf"
        bOK = (Right(strLine, 4) = "Then" Or Right(strLine, 1) = "_")
    Case "Private", "Public", "Friend"
        nPos = InStr(1, strLine, " Static ")
        If nPos Then
            nPos = InStr(nPos + 7, strLine, " ")
        Else
            nPos = InStr(Len(strTemp) + 1, strLine, " ")
        End If
        On Error GoTo Skip
        Select Case Mid$(strLine, nPos + 1, InStr(nPos + 1, strLine, " ") - nPos - 1)
        Case "Sub", "Function", "Property", "Enum", "Type"
            bOK = True
        End Select
        Skip:
        On Error GoTo 0
    End Select
    IsBlockStart = bOK
End Function
Private Sub ColorizeBlockLinksByLevel()
    Dim rnd As Long
    Dim n As Variant
    Dim i As Long
    Dim s As Shape
    Dim sNames As New Collection
    Set sNames = New Collection
    For Each s In ThisWorkbook.Sheets("ProjectManagerPrinter").Shapes
        If UCase(s.Name) <> "LOGO" And Not s.Name Like "Image*" Then
            s.Name = s.Left
            On Error Resume Next
            sNames.Add s.Name, s.Name
            On Error GoTo 0
        End If
    Next s
```

```
    For Each n In sNames
        rnd = RandomRGB
        For Each s In ThisWorkbook.Sheets("ProjectManagerPrinter").Shapes
            If UCase(s.Name) <> "LOGO" And Not s.Name Like "Image*" Then
                If s.Name = n Then
                    With s.Line
                        .ForeColor.RGB = rnd
                        .Weight = 1.5
                    End With
                End If
            End If
        Next s
    Next n
    Set sNames = Nothing
End Sub
Private Function openPair(strLine As String) As String
    Dim nPos As Integer
    Dim strTemp As String
    strTemp = Trim(strLine)
    If strTemp Like "*Declare*" Then GoTo Skip
    nPos = InStr(1, strTemp, " ") - 1
    If nPos < 0 Then nPos = Len(strLine)
    strTemp = Left$(strLine, nPos)
    Select Case strTemp
    Case Is = "Private", "Public"
        strTemp = Trim(strLine)
        strTemp = Replace(strTemp, "Private ", "")
        strTemp = Replace(strTemp, "Public ", "")
        nPos = InStr(1, strTemp, " ") - 1
        If nPos < 0 Then nPos = Len(strTemp)
        strTemp = Left$(strTemp, nPos)
        If strTemp = "Function" Then
            openPair = "Function"
        ElseIf strTemp = "Sub" Then
            openPair = "Sub"
        Else
            GoTo Skip
        End If
    Case Is = "Sub"
        openPair = "Sub"
    Case Is = "Function"
        openPair = "Function"
    Case Is = "With"
        openPair = "With"
    Case Is = "For"
        openPair = "For"
    Case Is = "Do"
        openPair = "Do"
    Case Is = "While"
        openPair = "While"
    Case Is = "Select"
        openPair = "Select"
    Case Is = "Case"
        openPair = "Case"
    Case Is = "Property"
        openPair = "Property"
    Case Is = "Enum"
        openPair = "Enum"
    Case Is = "Type"
```

```
        openPair = "Type"
    Case "If"
        openPair = "If"
    Case "ElseIf", "Else", "Else:"
        openPair = "Else"
    Case "#If"
        openPair = "#If"
    Case "#ElseIf", "#Else", "#Else:"
        openPair = "#Else"
    Case Else
        Skip:
        openPair = "skip"
    End Select
End Function
Private Function closePair(strLine As String) As String
    Dim nPos As Integer
    Dim strTemp As String
    nPos = InStr(1, strLine, " ") - 1
    If nPos < 0 Then nPos = Len(strLine)
    strTemp = Left$(strLine, nPos)
    Select Case strTemp
    Case Is = "Private", "Public"
        strTemp = Trim(strLine)
        strTemp = Replace(strTemp, "Private ", "")
        strTemp = Replace(strTemp, "Public ", "")
        nPos = InStr(1, strTemp, " ") - 1
        If nPos < 0 Then nPos = Len(strTemp)
        strTemp = Left$(strTemp, nPos)
        If strTemp = "Function" Then
            closePair = "End Function"
        ElseIf strTemp = "Sub" Then
            closePair = "End Sub"
        Else
        End If
    Case Is = "Sub"
        closePair = "End Sub"
    Case Is = "Function"
        closePair = "End Function"
    Case Is = "With"
        closePair = "End With"
    Case Is = "For"
        closePair = "Next"
    Case Is = "Do"
        closePair = "Loop"
    Case Is = "While"
        closePair = "Wend"
    Case Is = "Select", "Case"
        closePair = "End Select"
    Case Is = "Property"
        closePair = "End Property"
    Case Is = "Enum"
        closePair = "End Enum"
    Case Is = "Type"
        closePair = "End Type"
    Case "If", "ElseIf", "Else", "Else:"
        closePair = "End If"
    Case "#If", "#ElseIf", "#Else", "#Else:"
        closePair = "#End If"
    Case Else
```

```
    End Select
End Function
Private Function AddShape() As Shape
    Dim shp As Shape
    Set shp = ActiveSheet.Shapes.AddShape _
    (msoShapeRoundedRectangle, 1, 1, 500, 10)
    With shp.ThreeD
        .BevelTopType = msoBevelCircle
        .BevelTopInset = 6
        .BevelTopDepth = 6
    End With
    With shp.Fill
        .visible = msoTrue
        .ForeColor.RGB = RGB(0, 176, 80)
        .Transparency = 0
        .Solid
    End With
    With shp.Line
        .visible = msoTrue
        .ForeColor.ObjectThemeColor = msoThemeColorBackground1
        .ForeColor.TintAndShade = 0
        .ForeColor.Brightness = 0
        .Transparency = 0
    End With
    Set AddShape = shp
End Function
Private Function StrWidth(s As String, sFontName As String, fFontSize As Double) As Double
    Dim i As Long
    Dim j As Long
    If sFontName <> msFontName Then
        If Not InitChrWidths(sFontName) Then
            Exit Function
        End If
    End If
    For i = 1 To Len(s)
        j = Asc(Mid(s, i, 1))
        If j >= 32 Then
            StrWidth = StrWidth + fFontSize * mafChrWid(j)
        End If
    Next i
End Function
Private Function getLastColumn(TargetSheet As Worksheet) As Long
    Dim LastCell As Range
    On Error Resume Next
    Set LastCell = ActiveSheet.Cells.Find("*", SearchOrder:=xlByColumns, SearchDirection:= _
    xlPrevious)
    On Error GoTo 0
    If LastCell Is Nothing Then
        getLastColumn = 1
    Else
        getLastColumn = LastCell.Column
    End If
End Function
Private Function FindAll(SearchRange As Range, _
    FindWhat As Variant, _
    Optional LookIn As XlFindLookIn = xlValues, _
    Optional LookAt As XlLookAt = xlWhole, _
    Optional SearchOrder As XlSearchOrder = xlByRows, _
    Optional MatchCase As Boolean = False, _
```

```
    Optional BeginsWith As String = vbNullString, _
    Optional EndsWith As String = vbNullString, _
    Optional BeginEndCompare As VbCompareMethod = vbTextCompare) As Range
    Dim FoundCell As Range
    Dim FirstFound As Range
    Dim LastCell As Range
    Dim ResultRange As Range
    Dim XLookAt As XlLookAt
    Dim Include As Boolean
    Dim CompMode As VbCompareMethod
    Dim Area As Range
    Dim maxRow As Long
    Dim MaxCol As Long
    Dim BeginB As Boolean
    Dim EndB As Boolean
    CompMode = BeginEndCompare
    If BeginsWith <> vbNullString Or EndsWith <> vbNullString Then
        XLookAt = xlPart
    Else
        XLookAt = LookAt
    End If
    For Each Area In SearchRange.Areas
        With Area
            If .Cells(.Cells.count).Row > maxRow Then
                maxRow = .Cells(.Cells.count).Row
            End If
            If .Cells(.Cells.count).Column > MaxCol Then
                MaxCol = .Cells(.Cells.count).Column
            End If
        End With
    Next Area
    Set LastCell = SearchRange.Worksheet.Cells(maxRow, MaxCol)
    On Error GoTo 0
    Set FoundCell = SearchRange.Find(what:=FindWhat, _
    After:=LastCell, _
    LookIn:=LookIn, _
    LookAt:=XLookAt, _
    SearchOrder:=SearchOrder, _
    MatchCase:=MatchCase)
    If Not FoundCell Is Nothing Then
        Set FirstFound = FoundCell
        Do Unti
            Include = False
            If BeginsWith = vbNullString And EndsWith = vbNullString Then
                Include = True
            Else
                If BeginsWith <> vbNullString Then
                    If StrComp(Left(FoundCell.TEXT, Len(BeginsWith)), BeginsWith, BeginEndCompare) =
                    0 Then
                        Include = True
                    End If
                End If
                If EndsWith <> vbNullString Then
                    If StrComp(Right(FoundCell.TEXT, Len(EndsWith)), EndsWith, BeginEndCompare) = 0
                    Then
                        Include = True
                    End If
                End If
            End If
```

```
            If Include = True Then
                If ResultRange Is Nothing Then
                    Set ResultRange = FoundCell
                Else
                    Set ResultRange = Application.Union(ResultRange, FoundCell)
                End If
            End If
            Set FoundCell = SearchRange.FindNext(After:=FoundCell)
            If (FoundCell Is Nothing) Then
                Exit Do
            End If
            If (FoundCell.Address = FirstFound.Address) Then
                Exit Do
            End If
        Loop
    End If
    Set FindAll = ResultRange
End Function
Private Sub printRange(var As Variant)
    If var.Areas.count = 1 Then
        dp var.Value
    Else
        Dim out As Variant
        Dim Temp As Variant
        Dim i As Long
        For i = 1 To var.Areas.count
            Temp = var.Areas(i).Value
            If IsEmpty(out) Then
                out = Temp
            Else
                out = Combine2Array(out, Temp)
            End If
        Next
        dp out
    End If
End Sub
Private Sub ColorWords(TargetRange As Range, ByRef MyWords As Variant, ByRef myColor As Long)
    Dim MatchPosition As Long
    Dim MyPattern As String
    Dim MyCell As Range
    Dim MyObj As Object
    MyPattern = Join$(MyWords, ",")
    With CreateObject("VBScript.RegExp")
        .Global = True
        .IgnoreCase = True
        .pattern = "([\^\$\(\)\[\]\*\+\-\?\.\|])"
        MyPattern = Replace(.Replace(MyPattern, "\$1"), ",", "|")
        .pattern = "\b(" & MyPattern & ")\b"
        For Each MyCell In TargetRange
            If .test(MyCell.Value) Then
                For Each MyObj In .Execute(MyCell.Value)
                    MatchPosition = WorksheetFunction.Match(MyObj, MyWords, 0)
                    If Not IsError(MatchPosition) Then
                        With MyCell.Characters(MyObj.FirstIndex + 1, MyObj.Length).Font
                            .color = myColor
                            .Bold = True
                        End With
                    End If
                Next
```

```
            End If
        Next MyCell
    End With
End Sub
Private Sub GreenifyComments()
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Sheets("ProjectManagerPrinter")
    Dim InlineComments As Range
    Dim rng As Range
    Dim cell As Range
    For Each cell In ws.Cells.SpecialCells(xlCellTypeConstants)
        If LTrim(cell.Value) Like "'*" Or Trim(cell.Value) = "Rem *" Then
            If rng Is Nothing Then
                Set rng = cell
            Else
                Set rng = Union(rng, cell)
            End If
        ElseIf CountOfCharacters(cell.TEXT, "'") = 1 Then
            If InlineComments Is Nothing Then
                Set InlineComments = cell
            Else
                Set InlineComments = Union(InlineComments, cell)
            End If
        End If
    Next
    If Not rng Is Nothing Then rng.Font.color = ws.Range("J4").Value
    If Not InlineComments Is Nothing Then GreenifyInlineComments InlineComments
End Sub
Private Function RangeToString(ByVal myRange As Range, Optional delim As String = ",") As String
    RangeToString = ""
    If Not myRange Is Nothing Then
        Dim MyCell As Range
        For Each MyCell In myRange
            RangeToString = RangeToString & delim & MyCell.Value
        Next MyCell
        RangeToString = Right(RangeToString, Len(RangeToString) - Len(delim))
    End If
End Function
Private Sub DeleteWorksheet(TargetWorksheet As Worksheet)
    Application.DisplayAlerts = False
    TargetWorksheet.Delete
    Application.DisplayAlerts = True
End Sub
Private Sub PrintScreen()
    keybd_event VK_LMENU, 0, KEYEVENTF_EXTENDEDKEY, 0
    keybd_event VK_SNAPSHOT, 0, KEYEVENTF_EXTENDEDKEY, 0
    keybd_event VK_SNAPSHOT, 0, KEYEVENTF_EXTENDEDKEY + KEYEVENTF_KEYUP, 0
    keybd_event VK_LMENU, 0, KEYEVENTF_EXTENDEDKEY + KEYEVENTF_KEYUP, 0
End Sub
Private Function ClearClipboard()
    OpenClipboard (0&)
    EmptyClipboard
    CloseClipboard
End Function
Private Sub SetVbeOnTop()
    ShowOnTop Application.VBE.MainWindow.hWnd, True
End Sub
Private Sub SetVbeNormal()
    ShowOnTop Application.VBE.MainWindow.hWnd, False
```

```
End Sub
Private Sub SetXLOnTop()
    ShowOnTop Application.hWnd, True
End Sub
Private Function getWhichFirstPosition(st As String, items As Variant, BeforeOrAfterPosition As _
Long, lookForward As Boolean)
    Dim i As Long
    Dim el As Variant
    getWhichFirstPosition = -1
    If lookForward Then
        For i = BeforeOrAfterPosition To Len(st)
            For Each el In items
                If Mid(st, i, 1) = el Then
                    getWhichFirstPosition = i
                    Exit Function
                End If
            Next
        Next
    Else
        For i = BeforeOrAfterPosition To 1 Step -1
            For Each el In items
                If Mid(st, i, 1) = el Then
                    getWhichFirstPosition = i
                    Exit Function
                End If
            Next
        Next i
    End If
End Function
Private Function RandomRGB()
    RandomRGB = RGB(Int(rnd() * 255), Int(rnd() * 255), Int(rnd() * 255))
End Function
Private Sub GreenifyInlineComments(rng As Range)
    Dim n               As Long
    Dim i               As Long
    Dim j               As Long
    Dim k               As Long
    Dim l               As Long
    Dim LineText        As String
    Dim exitString      As String
    Dim QUOTES          As Long
    Dim Q               As Long
    Dim StartPos        As Long
    For Each cell In rng
        LineText = Trim(cell.TEXT)
        StartPos = 1
        RETRY:
        n = InStr(StartPos, LineText, "'")
        Q = InStr(StartPos, LineText, """")
        QUOTES = 0
        If Q < n Then
            For l = 1 To n
                If Mid(LineText, l, 1) = """" Then
                    QUOTES = QUOTES + 1
                End If
            Next l
        End If
        If QUOTES = Application.WorksheetFunction.Odd(QUOTES) Then
            StartPos = n + 1
```

```
            GoTo RETRY:
        Else
            Select Case n
            Case Is = 0
            Case Is = 1
            Case Is > 1
                cell.Characters(n).Font.color = ws.Range("J4").Value
            End Select
        End If
    Next
End Sub
Private Function CountOfCharacters(SearchInString As String, FindWhat As Variant)
    If Not IsArray(FindWhat) Then FindWhat = Array(FindWhat)
    If UBound(FindWhat) = 0 And Len(FindWhat(0)) = 1 Then
        CountOfCharacters = Len(SearchInString) - Len(Replace(SearchInString, CStr(FindWhat(0)), "")
        )
    Else
        Dim counter As Long, total As Long
        Dim matches As New Collection
        For i = LBound(FindWhat) To UBound(FindWhat)
            If IsOnlyLetters(CStr(FindWhat(i))) Then
                If InStrExact(1, SearchInString, CStr(FindWhat(i))) > 0 Then
                    counter = 0
                    Do While (InStr(1, SearchInString, CStr(FindWhat(i)), vbTextCompare)) > 0
                        counter = counter + 1
                        SearchInString = Replace(SearchInString, FindWhat(i), "", , 1,  _
                        vbTextCompare)
                    Loop
                    matches.Add FindWhat(i) & " (" & counter & ")"
                    total = total + counter
                End If
            Else
                counter = counter + 1
                total = total + counter
                matches.Add FindWhat(i) & " (" & counter & ")"
            End If
        Next
        CountOfCharacters = total
        dp matches
    End If
    Set matches = Nothing
End Function
Private Sub dp(var As Variant)
    Dim element     As Variant
    Dim i As Long
    Select Case TypeName(var)
    Case Is = "String", "Long", "Integer", "Double", "Boolean"
        Debug.Print var
    Case Is = "Variant()", "String()", "Long()", "Integer()"
        printArray var
    Case Is = "Collection"
        printCollection var
    Case Is = "Dictionary"
        printDictionary var
    Case Is = "Range"
        printRange var
    Case Is = "Date"
        Debug.Print var
    Case Else
```

```
    End Select
End Sub
Private Function IsOnlyLetters(s$) As Boolean
    IsOnlyLetters = Not s Like "*[!a-zA-Z]*"
End Function
Private Function InStrExact(Start As Long, sourceText As String, WordToFind As String, _
    Optional CaseSensitive As Boolean = False, _
    Optional AllowAccentedCharacters As Boolean = False) As Long
    Dim X As Long, Str1 As String, Str2 As String, pattern As String
    Const UpperAccentsOnly As String = "???"
    Const UpperAndLowerAccents As String = "??????"
    If CaseSensitive Then
        Str1 = sourceText
        Str2 = WordToFind
        pattern = "[!A-Za-z0-9]"
        If AllowAccentedCharacters Then pattern = Replace(pattern, "!", "!" & UpperAndLowerAccents)
    Else
        Str1 = UCase(sourceText)
        Str2 = UCase(WordToFind)
        pattern = "[!A-Z0-9]"
        If AllowAccentedCharacters Then pattern = Replace(pattern, "!", "!" & UpperAccentsOnly)
    End If
    For X = Start To Len(Str1) - Len(Str2) + 1
        If Mid(" " & Str1 & " ", X, Len(Str2) + 2) Like pattern & Str2 & pattern _
            And Not Mid(Str1, X) Like Str2 & "'[" & Mid(pattern, 3) & "*" Then
            InStrExact = X
            Exit Function
        End If
    Next
End Function
Private Sub FoldersCreate(FolderPath As String)
    On Error Resume Next
    Dim individualFolders() As String
    Dim tempFolderPath As String
    Dim arrayElement As Variant
    individualFolders = Split(FolderPath, "\")
    For Each arrayElement In individualFolders
        tempFolderPath = tempFolderPath & arrayElement & "\"
        If FolderExists(tempFolderPath) = False Then
            MkDir tempFolderPath
        End If
    Next arrayElement
End Sub
Private Function TxtAppend(sFile As String, sText As String)
    On Error GoTo ERR_HANDLER
    Dim iFileNumber           As Integer
    iFileNumber = FreeFile
    Open sFile For Append As #iFileNumber
    Print #iFileNumber, sText
    Close #iFileNumber
    Exit_Err_Handler:
    Exit Function
    ERR_HANDLER:
    MsgBox "The following error has occurred" & vbCrLf & vbCrLf & _
    "Error Number: " & err.Number & vbCrLf & _
    "Error Source: Txt_Append" & vbCrLf & _
    "Error Description: " & err.Description & _
    Switch(Erl = 0, "", Erl <> 0, vbCrLf & "Line No: " & Erl) _
    , vbOKOnly + vbCritical, "An Error has Occurred!"
```

```
    GoTo Exit_Err_Handler
End Function
Private Sub ShowOnTop(targetHwnd, ByVal OnTop As Boolean)
    Dim xStype As Long
    #If Win64 Then
        Dim xHwnd As LongPtr
    #Else
        Dim xHwnd As Long
    #End If
    If OnTop Then
        xStype = HWND_TOPMOST
    Else
        xStype = HWND_NOTOPMOST
    End If
    Call SetWindowPos(targetHwnd, xStype, 0, 0, 0, 0, SWP_NOSIZE Or SWP_NOMOVE)
End Sub
Private Function LastCell(rng As Range, Optional booCol As Boolean, Optional onlyAfterFirstCell As _
Boolean) As Range
    Dim ws As Worksheet
    Set ws = rng.parent
    Dim cell As Range
    If booCol = False Then
        Set cell = ws.Cells(Rows.count, rng.Column).End(xlUp)
        If cell.MergeCells Then Set cell = Cells(cell.Row + cell.Rows.count - 1, cell.Column)
    Else
        Set cell = ws.Cells(rng.Row, Columns.count).End(xlToLeft)
        If cell.MergeCells Then Set cell = Cells(cell.Row, cell.Column + cell.Columns.count - 1)
    End If
    If onlyAfterFirstCell = True Then
        If booCol = False Then
            Do While cell.Row <= rng.Row
                Set cell = cell.OFFSET(1, 0)
            Loop
        Else
            Do While cell.Column <= rng.Column
                Set cell = cell.OFFSET(0, 1)
            Loop
        End If
    End If
    Set LastCell = cell
End Function
Private Function IsBlockEnd(strLine As String) As Boolean
    strLine = Replace(strLine, Chr(13), "")
    Dim bOK As Boolean
    Dim nPos As Integer
    Dim strTemp As String
    nPos = InStr(1, strLine, " ") - 1
    If nPos < 0 Then nPos = Len(strLine)
    strTemp = Left$(strLine, nPos)
    Select Case strTemp
    Case "Next", "Loop", "Wend", "Case", "Else", "#Else", "Else:", "#Else:", "ElseIf", "#ElseIf", _
    "#End"
        bOK = True
    Case "End"
        bOK = (Len(strLine) > 3)
    End Select
    IsBlockEnd = bOK
End Function
Private Function MoveCommentQuoteToActualPosition(str As Variant)
```

```
    Dim var As Variant
    ReDim var(0)
    If TypeName(str) Like "String*" Then str = Split(str, vbNewLine)
    For j = LBound(str) To UBound(str)
        LineText = Trim(str(j))
        If Left(LineText, 2) = "' " Then
            tmp = Mid(LineText, 2)
            dif = Len(tmp) - Len(LTrim(tmp))
            var(UBound(var)) = Space(dif) & "'" & LTrim(tmp)
            ReDim Preserve var(UBound(var) + 1)
        Else
            var(UBound(var)) = str(j)
            ReDim Preserve var(UBound(var) + 1)
        End If
    Next
    MoveCommentQuoteToActualPosition = Join(var, vbNewLine)
    MoveCommentQuoteToActualPosition = Left(MoveCommentQuoteToActualPosition, Len( _
    MoveCommentQuoteToActualPosition) - Len(vbNewLine))
End Function
Private Function GetCompText(Module As Variant) As String
    If TypeName(Module) <> "VBComponent" Then Stop
    If Module.CodeModule.CountOfLines = 0 Then
        GetCompText = ""
        Exit Function
    End If
    GetCompText = Module.CodeModule.Lines(1, Module.CodeModule.CountOfLines)
End Function
Private Function ActiveCodepaneWorkbook() As Workbook
    Dim tmpstr As String
    tmpstr = Application.VBE.SelectedVBComponent.Collection.parent.fileName
    tmpstr = Right(tmpstr, Len(tmpstr) - InStrRev(tmpstr, "\"))
    Set ActiveCodepaneWorkbook = Workbooks(tmpstr)
End Function
Private Function GetProjectText(TargetWorkbook) As String
    If TypeName(TargetWorkbook) <> "Workbook" Then Stop
    Dim Module As VBComponent
    Dim txt
    Dim div As String
    div = vbNewLine & "'=========================================" & vbNewLine
    For Each Module In TargetWorkbook.VBProject.VBComponents
        If Module.CodeModule.CountOfLines > 0 Then
            txt = txt & div & _
            "'    " & getModuleName(Module) & "    " & Module.Type & _
            div & _
            GetModuleText(Module)
        End If
    Next
    GetProjectText = txt
End Function
Private Function GetProcText(vbComp As VBComponent, sProcName As Variant, _
    Optional bInclHeader As Boolean = True) As String
    If vbComp Is Nothing Then
        Stop
    End If
    Dim CodeMod As CodeModule
    Set CodeMod = vbComp.CodeModule
    Dim lProcStart          As Long
    Dim lProcBodyStart      As Long
    Dim lProcNoLines        As Long
```

```
    Const vbext_pk_Proc = 0
    On Error GoTo Error_Handler
    lProcStart = CodeMod.ProcStartLine(sProcName, vbext_pk_Proc)
    lProcBodyStart = CodeMod.ProcBodyLine(sProcName, vbext_pk_Proc)
    lProcNoLines = CodeMod.ProcCountLines(sProcName, vbext_pk_Proc)
    If bInclHeader = True Then
        GetProcText = CodeMod.Lines(lProcStart, lProcNoLines)
    Else
        lProcNoLines = lProcNoLines - (lProcBodyStart - lProcStart)
        GetProcText = CodeMod.Lines(lProcBodyStart, lProcNoLines)
    End If
    Error_Handler_Exit:
    On Error Resume Next
    Exit Function
    Error_Handler:
    "Error Source: GetProcText" & vbCrLf & _
    "Error Description: " & err.Description & _
    Switch(Erl = 0, vbNullString, Erl <> 0, vbCrLf & "Line No: " & Erl)
    Resume Error_Handler_Exit
End Function
Private Function ModuleOfProcedure(wb As Workbook, ProcedureName As Variant) As VBComponent
    Dim ProcKind As VBIDE.vbext_ProcKind
    Dim LineNum As Long, NumProc As Long
    Dim procName As String
    Dim vbComp As VBComponent
    For Each vbComp In wb.VBProject.VBComponents
        With vbComp.CodeModule
            LineNum = .CountOfDeclarationLines + 1
            Do Until LineNum >= .CountOfLines
                procName = .ProcOfLine(LineNum, ProcKind)
                LineNum = .ProcStartLine(procName, ProcKind) + .ProcCountLines(procName, ProcKind)  _

                If UCase(procName) = UCase(ProcedureName) Then
                    Set ModuleOfProcedure = vbComp
                    Exit Function
                End If
            Loop
        End With
    Next vbComp
End Function
Private Function getModuleName(Module As VBComponent) As String
    If Module.Type = vbext_ct_Document Then
        If Module.Name = "ThisWorkbook" Then
            getModuleName = Module.Name
        Else
            getModuleName = GetSheetByCodeName(WorkbookOfModule(Module), Module.Name).Name
        End If
    Else
        getModuleName = Module.Name
    End If
End Function
Private Function GetModuleText(vbComp As VBComponent) As String
    Dim CodeMod As CodeModule
    Set CodeMod = vbComp.CodeModule
    If CodeMod.CountOfLines = 0 Then GetModuleText = "": Exit Function
    GetModuleText = CodeMod.Lines(1, CodeMod.CountOfLines)
End Function
Private Function WorkbookOfModule(vbComp As VBComponent) As Workbook
    Set WorkbookOfModule = WorkbookOfProject(vbComp.Collection.parent)
```

```
End Function
Private Function WorkbookOfProject(vbProj As VBProject) As Workbook
    tmpstr = vbProj.fileName
    tmpstr = Right(tmpstr, Len(tmpstr) - InStrRev(tmpstr, "\"))
    Set WorkbookOfProject = Workbooks(tmpstr)
End Function
Private Function ArrayDimensionLength(SourceArray As Variant) As Integer
    Dim i As Integer
    Dim test As Long
    On Error GoTo catch
    Do
        i = i + 1
        test = UBound(SourceArray, i)
    Loop
    catch:
    ArrayDimensionLength = i - 1
End Function
Private Function InitChrWidths(sFontName As String) As Boolean
    Dim i As Long
    Select Case sFontName
    Case "Consolas"
        For i = 32 To 127
            Select Case i
            Case 32 To 127
                mafChrWid(i) = 0.5634
            End Select
        Next i
    Case "Arial"
        For i = 32 To 127
            Select Case i
            Case 39, 106, 108
                mafChrWid(i) = 0.1902
            Case 105, 116
                mafChrWid(i) = 0.2526
            Case 32, 33, 44, 46, 47, 58, 59, 73, 91 To 93, 102, 124
                mafChrWid(i) = 0.3144
            Case 34, 40, 41, 45, 96, 114, 123, 125
                mafChrWid(i) = 0.3768
            Case 42, 94, 118, 120
                mafChrWid(i) = 0.4392
            Case 107, 115, 122
                mafChrWid(i) = 0.501
            Case 35, 36, 48 To 57, 63, 74, 76, 84, 90, 95, 97 To 101, 103, 104, 110 To 113, 117, 121
                mafChrWid(i) = 0.5634
            Case 43, 60 To 62, 70, 126
                mafChrWid(i) = 0.6252
            Case 38, 65, 66, 69, 72, 75, 78, 80, 82, 83, 85, 86, 88, 89, 119
                mafChrWid(i) = 0.6876
            Case 67, 68, 71, 79, 81
                mafChrWid(i) = 0.7494
            Case 77, 109, 127
                mafChrWid(i) = 0.8118
            Case 37
                mafChrWid(i) = 0.936
            Case 64, 87
                mafChrWid(i) = 1.0602
            End Select
        Next i
    Case "Calibri"
```

```
        For i = 32 To 127
            Select Case i
            Case 32, 39, 44, 46, 73, 105, 106, 108
                mafChrWid(i) = 0.2526
            Case 40, 41, 45, 58, 59, 74, 91, 93, 96, 102, 123, 125
                mafChrWid(i) = 0.3144
            Case 33, 114, 116
                mafChrWid(i) = 0.3768
            Case 34, 47, 76, 92, 99, 115, 120, 122
                mafChrWid(i) = 0.4392
            Case 35, 42, 43, 60 To 63, 69, 70, 83, 84, 89, 90, 94, 95, 97, 101, 103, 107, 118, 121,
            124, 126
                mafChrWid(i) = 0.501
            Case 36, 48 To 57, 66, 67, 75, 80, 82, 88, 98, 100, 104, 110 To 113, 117, 127
                mafChrWid(i) = 0.5634
            Case 65, 68, 86
                mafChrWid(i) = 0.6252
            Case 71, 72, 78, 79, 81, 85
                mafChrWid(i) = 0.6876
            Case 37, 38, 119
                mafChrWid(i) = 0.7494
            Case 109
                mafChrWid(i) = 0.8742
            Case 64, 77, 87
                mafChrWid(i) = 0.936
            End Select
        Next i
    Case "Tahoma"
        For i = 32 To 127
            Select Case i
            Case 39, 105, 108
                mafChrWid(i) = 0.2526
            Case 32, 44, 46, 102, 106
                mafChrWid(i) = 0.3144
            Case 33, 45, 58, 59, 73, 114, 116
                mafChrWid(i) = 0.3768
            Case 34, 40, 41, 47, 74, 91 To 93, 124
                mafChrWid(i) = 0.4392
            Case 63, 76, 99, 107, 115, 118, 120 To 123, 125
                mafChrWid(i) = 0.501
            Case 36, 42, 48 To 57, 70, 80, 83, 95 To 98, 100, 101, 103, 104, 110 To 113, 117
                mafChrWid(i) = 0.5634
            Case 66, 67, 69, 75, 84, 86, 88, 89, 90
                mafChrWid(i) = 0.6252
            Case 38, 65, 71, 72, 78, 82, 85
                mafChrWid(i) = 0.6876
            Case 35, 43, 60 To 62, 68, 79, 81, 94, 126
                mafChrWid(i) = 0.7494
            Case 77, 119
                mafChrWid(i) = 0.8118
            Case 109
                mafChrWid(i) = 0.8742
            Case 64, 87
                mafChrWid(i) = 0.936
            Case 37, 127
                mafChrWid(i) = 1.0602
            End Select
        Next i
    Case "Lucida Console"
```

```
        For i = 32 To 127
            Select Case i
            Case 32 To 127
                mafChrWid(i) = 0.6252
            End Select
        Next i
    Case "Times New Roman"
        For i = 32 To 127
            Select Case i
            Case 39, 124
                mafChrWid(i) = 0.1902
            Case 32, 44, 46, 59
                mafChrWid(i) = 0.2526
            Case 33, 34, 47, 58, 73, 91 To 93, 105, 106, 108, 116
                mafChrWid(i) = 0.3144
            Case 40, 41, 45, 96, 102, 114
                mafChrWid(i) = 0.3768
            Case 63, 74, 97, 115, 118, 122
                mafChrWid(i) = 0.4392
            Case 94, 98 To 101, 103, 104, 107, 110, 112, 113, 117, 120, 121, 123, 125
                mafChrWid(i) = 0.501
            Case 35, 36, 42, 48 To 57, 70, 83, 84, 95, 111, 126
                mafChrWid(i) = 0.5634
            Case 43, 60 To 62, 69, 76, 80, 90
                mafChrWid(i) = 0.6252
            Case 65 To 67, 82, 86, 89, 119
                mafChrWid(i) = 0.6876
            Case 68, 71, 72, 75, 78, 79, 81, 85, 88
                mafChrWid(i) = 0.7494
            Case 38, 109, 127
                mafChrWid(i) = 0.8118
            Case 37
                mafChrWid(i) = 0.8742
            Case 64, 77
                mafChrWid(i) = 0.936
            Case 87
                mafChrWid(i) = 0.9984
            End Select
        Next i
    Case Else
        MsgBox "Font name """ & sFontName & """ not available!", vbCritical, "StrWidth"
        Exit Function
    End Select
    msFontName = sFontName
    InitChrWidths = True
End Function
Private Function RangeFindAll(ByRef SearchRange As Range, ByVal Value As Variant, Optional ByVal _
LookIn As XlFindLookIn = xlValues, Optional LookAt As XlLookAt = xlPart) As Range
    Dim FoundValues As Range, found As Range, Prev As Range, Looper As Boolean: Looper = True
    Do While Looper
        If Not Prev Is Nothing Then Set found = SearchRange.Find(Value, Prev, LookIn, LookAt)
        If found Is Nothing Then Set found = SearchRange.Find(Value, LookIn:=LookIn, LookAt:=LookAt)
        If Not found Is Nothing Then
            If FoundValues Is Nothing Then
                Set FoundValues = found
            Else
                If Not Intersect(found, FoundValues) Is Nothing Then Looper = False
                Set FoundValues = Union(FoundValues, found)
            End If
```

```
            Set Prev = found
        End If
        If found Is Nothing And Prev Is Nothing Then Exit Function
    Loop
    Set RangeFindAll = FoundValues
    Set FoundValues = Nothing
    Set found = Nothing
    Set Prev = Nothing
End Function
Private Function Combine2Array(ByVal arr1 As Variant, ByVal arr2 As Variant) As Variant
    Dim LowRowArr1 As Long
    Dim HighRowArr1 As Long
    Dim LowColumnArr1 As Long
    Dim HighColumnArr1 As Long
    Dim NumOfRowsArr1 As Long
    Dim NumOfColumnsArr1 As Long
    Dim LowRowArr2 As Long
    Dim HighRowArr2 As Long
    Dim LowColumnArr2 As Long
    Dim HighColumnArr2 As Long
    Dim NumOfRowsArr2 As Long
    Dim NumOfColumnsArr2 As Long
    Dim output As Variant
    Dim OutputRow As Long
    Dim OutputColumn As Long
    Dim RowIdx As Long
    Dim ColIdx As Long
    If (IsArray(arr1) = False) Or (IsArray(arr2) = False) Then
        Combine2Array = Null
        MsgBox "Both need to be array"
        Exit Function
    End If
    If (NumberOfArrayDimensions(arr1) <> 2) Or (NumberOfArrayDimensions(arr2) <> 2) Then
        Combine2Array = Null
        MsgBox "Both need to be 2D array"
        Exit Function
    End If
    LowRowArr1 = LBound(arr1, 1)
    HighRowArr1 = UBound(arr1, 1)
    LowColumnArr1 = LBound(arr1, 2)
    HighColumnArr1 = UBound(arr1, 2)
    NumOfRowsArr1 = HighRowArr1 - LowRowArr1 + 1
    NumOfColumnsArr1 = HighColumnArr1 - LowColumnArr1 + 1
    LowRowArr2 = LBound(arr2, 1)
    HighRowArr2 = UBound(arr2, 1)
    LowColumnArr2 = LBound(arr2, 2)
    HighColumnArr2 = UBound(arr2, 2)
    NumOfRowsArr2 = HighRowArr2 - LowRowArr2 + 1
    NumOfColumnsArr2 = HighColumnArr2 - LowColumnArr2 + 1
    If NumOfColumnsArr1 <> NumOfColumnsArr2 Then
        Combine2Array = Null
        MsgBox "Both array must have same number of column"
        Exit Function
    End If
    ReDim output(0 To NumOfRowsArr1 + NumOfRowsArr2 - 1, 0 To NumOfColumnsArr1 - 1)
    For RowIdx = LowRowArr1 To HighRowArr1
        OutputColumn = 0
        For ColIdx = LowColumnArr1 To HighColumnArr1
            output(OutputRow, OutputColumn) = arr1(RowIdx, ColIdx)
```

```
            OutputColumn = OutputColumn + 1
        Next ColIdx
        OutputRow = OutputRow + 1
    Next RowIdx
    For RowIdx = LowRowArr2 To HighRowArr2
        OutputColumn = 0
        For ColIdx = LowColumnArr2 To HighColumnArr2
            output(OutputRow, OutputColumn) = arr2(RowIdx, ColIdx)
            OutputColumn = OutputColumn + 1
        Next ColIdx
        OutputRow = OutputRow + 1
    Next RowIdx
    Combine2Array = output
End Function
Private Sub printArray(var As Variant)
    If ArrayDimensions(var) = 1 Then
        Debug.Print Join(var, vbNewLine)
    ElseIf ArrayDimensions(var) > 1 Then
        DPH var
    End If
End Sub
Private Sub printCollection(var As Variant)
    Dim elem        As Variant
    For Each elem In var
        dp elem
    Next elem
End Sub
Private Sub printDictionary(var As Variant)
    Dim i As Long: Dim iCount As Long
    Dim arrKeys
    Dim sKey        As String
    Dim varItem
    With var
        iCount = .count
        arrKeys = .keys
        iCount = UBound(arrKeys, 1)
        For i = 0 To iCount
            sKey = arrKeys(i)
            If IsObject(.Item(sKey)) Then
                Debug.Print sKey & " : "
                dp (.Item(sKey))
            Else
                Debug.Print sKey & " : " & .Item(sKey)
            End If
        Next i
    End With
End Sub
Private Sub DPH(ByVal Hairetu, Optional HyoujiMaxNagasa%, Optional HairetuName$)
    Call DebugPrintHairetu(Hairetu, HyoujiMaxNagasa, HairetuName)
End Sub
Private Function ArrayDimensions(ByVal vArray As Variant) As Long
    Dim dimnum      As Long
    Dim ErrorCheck As Long
    On Error GoTo FinalDimension
    For dimnum = 1 To 60000
        ErrorCheck = LBound(vArray, dimnum)
    Next
    FinalDimension:
    ArrayDimensions = dimnum - 1
```

```
End Function
Private Sub DebugPrintHairetu(ByVal Hairetu, Optional HyoujiMaxNagasa%, Optional HairetuName$)
    Dim i&, j&, k&, M&, n&
    Dim TateMin&, TateMax&, YokoMin&, YokoMax&
    Dim WithTableHairetu
    Dim NagasaList, MaxNagasaList
    Dim NagasaOnajiList
    Dim OutputList
    Const SikiriMoji$ = "|"
    Dim Jigen2%
    On Error Resume Next
    Jigen2 = UBound(Hairetu, 2)
    On Error GoTo 0
    If Jigen2 = 0 Then
        Hairetu = Application.Transpose(Hairetu)
    End If
    TateMin = LBound(Hairetu, 1)
    TateMax = UBound(Hairetu, 1)
    YokoMin = LBound(Hairetu, 2)
    YokoMax = UBound(Hairetu, 2)
    ReDim WithTableHairetu(1 To TateMax - TateMin + 1 + 1, 1 To YokoMax - YokoMin + 1 + 1)
    For i = 1 To TateMax - TateMin + 1
        WithTableHairetu(i + 1, 1) = TateMin + i - 1
        For j = 1 To YokoMax - YokoMin + 1
            WithTableHairetu(1, j + 1) = YokoMin + j - 1
            WithTableHairetu(i + 1, j + 1) = Hairetu(i - 1 + TateMin, j - 1 + YokoMin)
        Next j
    Next i
    n = UBound(WithTableHairetu, 1)
    M = UBound(WithTableHairetu, 2)
    ReDim NagasaList(1 To n, 1 To M)
    ReDim MaxNagasaList(1 To M)
    Dim tmpstr$
    For j = 1 To M
        For i = 1 To n
            If j > 1 And HyoujiMaxNagasa <> 0 Then
                tmpstr = WithTableHairetu(i, j)
                WithTableHairetu(i, j) = ShortenToByteCharacters(tmpstr, HyoujiMaxNagasa)
            End If
            NagasaList(i, j) = LenB(StrConv(WithTableHairetu(i, j), vbFromUnicode))
            MaxNagasaList(j) = WorksheetFunction.Max(MaxNagasaList(j), NagasaList(i, j))
        Next i
    Next j
    ReDim NagasaOnajiList(1 To n, 1 To M)
    Dim TmpMaxNagasa&
    For j = 1 To M
        TmpMaxNagasa = MaxNagasaList(j)
        For i = 1 To n
            NagasaOnajiList(i, j) = WithTableHairetu(i, j) & WorksheetFunction.Rept(" ",  _
            TmpMaxNagasa - NagasaList(i, j))
        Next i
    Next j
    ReDim OutputList(1 To n)
    For i = 1 To n
        For j = 1 To M
            If j = 1 Then
                OutputList(i) = NagasaOnajiList(i, j)
            Else
                OutputList(i) = OutputList(i) & SikiriMoji & NagasaOnajiList(i, j)
```

```
            End If
        Next j
    Next i
    Debug.Print HairetuName
    For i = 1 To n
        Debug.Print OutputList(i)
    Next i
End Sub
Private Function ShortenToByteCharacters(Mojiretu$, ByteNum%)
    Dim OriginByte%
    Dim output
    OriginByte = LenB(StrConv(Mojiretu, vbFromUnicode))
    If OriginByte <= ByteNum Then
        output = Mojiretu
    Else
        Dim RuikeiByteList, BunkaiMojiretu
        RuikeiByteList = CalculateByteCharacters(Mojiretu)
        BunkaiMojiretu = TextDecomposition(Mojiretu)
        Dim AddMoji$
        AddMoji = "."
        Dim i&, n&
        n = Len(Mojiretu)
        For i = 1 To n
            If RuikeiByteList(i) < ByteNum Then
                output = output & BunkaiMojiretu(i)
            ElseIf RuikeiByteList(i) = ByteNum Then
                If LenB(StrConv(BunkaiMojiretu(i), vbFromUnicode)) = 1 Then
                    output = output & AddMoji
                Else
                    output = output & AddMoji & AddMoji
                End If
                Exit For
            ElseIf RuikeiByteList(i) > ByteNum Then
                output = output & AddMoji
                Exit For
            End If
        Next i
    End If
    ShortenToByteCharacters = output
End Function
Private Function CalculateByteCharacters(Mojiretu$)
    Dim MojiKosu%
    MojiKosu = Len(Mojiretu)
    Dim output
    ReDim output(1 To MojiKosu)
    Dim i&
    Dim TmpMoji$
    For i = 1 To MojiKosu
        TmpMoji = Mid(Mojiretu, i, 1)
        If i = 1 Then
            output(i) = LenB(StrConv(TmpMoji, vbFromUnicode))
        Else
            output(i) = LenB(StrConv(TmpMoji, vbFromUnicode)) + output(i - 1)
        End If
    Next i
    CalculateByteCharacters = output
End Function
Private Function TextDecomposition(Mojiretu$)
    Dim i&, n&
```

```
    Dim output
    n = Len(Mojiretu)
    ReDim output(1 To n)
    For i = 1 To n
        output(i) = Mid(Mojiretu, i, 1)
    Next i
    TextDecomposition = output
End Function
Private Function FolderExists(ByVal strPath As String) As Boolean
    On Error Resume Next
    FolderExists = ((GetAttr(strPath) And vbDirectory) = vbDirectory)
End Function
Private Function NumberOfArrayDimensions(Arr As Variant) As Byte
    Dim Ndx As Byte
    Dim Res As Long
    On Error Resume Next
    Do
        Ndx = Ndx + 1
        Res = UBound(Arr, Ndx)
    Loop Until err.Number <> 0
    NumberOfArrayDimensions = Ndx - 1
End Function
```

--- F_Userforms ---

```
Option Explicit
Option Compare Text
Private Declare PtrSafe Function DrawMenuBar Lib "user32" (ByVal hWnd As Long) As Long
Private Declare PtrSafe Function SetLayeredWindowAttributes Lib "user32" (ByVal hWnd As Long, ByVal
crKey As Long, ByVal bAlpha As Byte, ByVal dwFlags As Long) As Long
Private Const GWL_STYLE As Long = (-16)
Private Const GWL_EXSTYLE As Long = (-20)
Private Const WS_CAPTION As Long = &HC00000
Private Const WS_EX_DLGMODALFRAME As Long = &H1
Private Const WS_EX_LAYERED = &H80000
Private Const LWA_COLORKEY = &H1
Private Const LWA_ALPHA = &H2
Private m_sngDownX As Single
Private m_sngDownY As Single
Public Const FORMAT_MESSAGE_ALLOCATE_BUFFER = &H100
Public Const FORMAT_MESSAGE_ARGUMENT_ARRAY = &H2000
Public Const FORMAT_MESSAGE_FROM_HMODULE = &H800
Public Const FORMAT_MESSAGE_FROM_STRING = &H400
Public Const FORMAT_MESSAGE_FROM_SYSTEM = &H1000
Public Const FORMAT_MESSAGE_IGNORE_INSERTS = &H200
Public Const FORMAT_MESSAGE_MAX_WIDTH_MASK = &HFF
Public Const FORMAT_MESSAGE_TEXT_LEN = 160
Public Const MAX_PATH = 260
Public Const GWL_HWNDPARENT As Long = -8
Public Const GW_OWNER = 4
Public Declare PtrSafe Function GetWindowLong Lib "user32.dll" Alias "GetWindowLongA" (ByVal hWnd  _
As Long, ByVal nIndex As Long) As Long
Public Declare PtrSafe Function GetClassName Lib "user32" Alias "GetClassNameA" (ByVal hWnd As Long,
ByVal lpClassName As String, ByVal nMaxCount As Long) As Long
Private Declare PtrSafe Function GetWindowText Lib "user32.dll" Alias "GetWindowTextA" (ByVal hWnd  _
As Long, ByVal lpString As String, ByVal cch As Long) As Long
Public VBEditorHWnd As Long
Public ApplicationHWnd As Long
Public ExcelDeskHWnd As Long
Public ActiveWindowHWnd As Long
Public UserFormHWnd As Long
Public WindowsDesktopHWnd As Long
Public Const GA_ROOT As Long = 2
Public Const GA_ROOTOWNER As Long = 3
Public Const GA_PARENT As Long = 1
Public Declare PtrSafe Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal hWnd1 As  _
Long, ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long
Public Declare PtrSafe Function GetAncestor Lib "user32.dll" (ByVal hWnd As Long, ByVal gaFlags As  _
Long) As Long
Public Declare PtrSafe Function GetDesktopWindow Lib "user32" () As Long
Public Declare PtrSafe Function GetParent Lib "user32.dll" (ByVal hWnd As Long) As Long
Private Declare PtrSafe Function GetWindow Lib "user32" (ByVal hWnd As Long, ByVal wCmd As Long) As
Long
Public Declare PtrSafe Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hWnd As  _
Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
Public Const C_EXCEL_APP_WINDOWCLASS = "XLMAIN"
Public Const C_EXCEL_DESK_WINDOWCLASS = "XLDESK"
Public Const C_EXCEL_WINDOW_WINDOWCLASS = "EXCEL7"
Public Const USERFORM_WINDOW_CLASS = "ThunderDFrame"
Public Const C_VBA_USERFORM_WINDOWCLASS = "ThunderDFrame"
Public Const SWP_NOMOVE = &H2
Public Const SWP_NOSIZE = &H1
```

```
    Public Const HWND_TOP = 0
    Public Const HWND_BOTTOM = 1
    Public Const HWND_TOPMOST = -1
    Public Const HWND_NOTOPMOST = -2
    Public Declare PtrSafe Function SetWindowPos Lib "user32" (ByVal hWnd As LongPtr, ByVal  _
    hWndInsertAfter As LongPtr, ByVal X As LongPtr, ByVal Y As LongPtr, ByVal cx As LongPtr, ByVal cy  _
    As LongPtr, ByVal uFlags As LongPtr) As Long
    #If VBA7 Then
        Public Declare PtrSafe Function SetParent Lib "user32" ( _
        ByVal hwndChild As LongPtr, _
        ByVal hWndNewParent As LongPtr) As LongPtr
        Public Declare PtrSafe Function SetForegroundWindow Lib "user32" ( _
        ByVal hWnd As LongPtr) As Long
        Public Declare PtrSafe Function FindWindow Lib "user32" Alias "FindWindowA" ( _
        ByVal lpClassName As String, _
        ByVal lpWindowName As String) As LongPtr
        Public Declare PtrSafe Function FormatMessage Lib "kernel32" Alias "FormatMessageA" ( _
        ByVal dwFlags As Long _
        , lpSource As Any _
        , ByVal dwMessageId As Long _
        , ByVal dwLanguageId As Long _
        , ByVal lpBuffer As String _
        , ByVal nSize As Long _
        , Arguments As LongPtr) As Long
    #Else
        Public Declare  Function SetParent Lib "user32" ( _
        ByVal hWndChild As Long, _
        ByVal hWndNewParent As Long) As Long
        Public Declare  Function SetForegroundWindow Lib "user32" ( _
        ByVal hwnd As Long) As Long
        Public Declare  Function FindWindow Lib "user32" Alias "FindWindowA" ( _
        ByVal lpClassName As String, _
        ByVal lpWindowName As String) As Long
        Public Declare  Function FormatMessage Lib "kernel32.dll" Alias "FormatMessageA" ( _
        ByVal dwFlags As Long, _
        ByRef lpSource As Any, _
        ByVal dwMessageId As Long, _
        ByVal dwLanguageId As Long, _
        ByVal lpBuffer As String, _
        ByVal nSize As Long, _
        ByRef Arguments As Long) As Long
    #End If
    Public Enum CloseBy
        user = 0
        code = 1
        WindowsOS = 2
        TaskManager = 3
    End Enum
    Public Declare PtrSafe Function getTickCount Lib "kernel32" Alias "GetTickCount" () As Long
    Public Const Black As Long = &H80000012
    Public Const Red As Long = &HFF&
    Public Const ControlIDCheckBox = "Forms.CheckBox.1"
    Public Const ControlIDComboBox = "Forms.ComboBox.1"
    Public Const ControlIDCommandButton = "Forms.CommandButton.1"
    Public Const ControlIDFrame = "Forms.Frame.1"
    Public Const ControlIDImage = "Forms.Image.1"
    Public Const ControlIDLabel = "Forms.Label.1"
    Public Const ControlIDListBox = "Forms.ListBox.1"
    Public Const ControlIDMultiPage = "Forms.MultiPage.1"
```

```
    Public Const ControlIDOptionButton = "Forms.OptionButton.1"
    Public Const ControlIDScrollBar = "Forms.ScrollBar.1"
    Public Const ControlIDSpinButton = "Forms.SpinButton.1"
    Public Const ControlIDTabStrip = "Forms.TabStrip.1"
    Public Const ControlIDTextBox = "Forms.TextBox.1"
    Public Const ControlIDToggleButton = "Forms.ToggleButton.1"
    Private Declare PtrSafe Function FindWindowA Lib "user32" (ByVal lpClassName As String, ByVal  _
    lpWindowName As String) As Long
    Private Declare PtrSafe Function GetWindowLongA Lib "user32" (ByVal hWnd As Long, ByVal nIndex As  _
    Long) As Long
    Private Declare PtrSafe Function SetWindowLongA Lib "user32" (ByVal hWnd As Long, ByVal nIndex As  _
    Long, ByVal dwNewLong As Long) As Long
    Public Sub MakeFormTransparent(frm As Object, Optional color As Variant)
        Dim formhandle As Long
        Dim bytOpacity As Byte
        formhandle = CLng(FindWindow(vbNullString, frm.Caption))
        If IsMissing(color) Then color = vbWhite
        bytOpacity = 100
        SetWindowLong formhandle, GWL_EXSTYLE, GetWindowLong(formhandle, GWL_EXSTYLE) Or WS_EX_LAYERED
        frm.BackColor = color
        SetLayeredWindowAttributes formhandle, color, bytOpacity, LWA_COLORKEY
    End Sub
    Public Sub MakeFormBorderless(frm As Object)
        Dim lngWindow As Long
        Dim lFrmHdl As Long
        lFrmHdl = CLng(FindWindow(vbNullString, frm.Caption))
        lngWindow = GetWindowLong(lFrmHdl, GWL_STYLE)
        lngWindow = lngWindow And (Not WS_CAPTION)
        SetWindowLong lFrmHdl, GWL_STYLE, lngWindow
        lngWindow = GetWindowLong(lFrmHdl, GWL_EXSTYLE)
        lngWindow = lngWindow And Not WS_EX_DLGMODALFRAME
        SetWindowLong lFrmHdl, GWL_EXSTYLE, lngWindow
        DrawMenuBar lFrmHdl
    End Sub
    Public Sub UserformOnTop(form As Object)
        Const C_VBA6_USERFORM_CLASSNAME = "ThunderDFrame"
        Dim ret As Long
        Dim formHWnd As Long
        formHWnd = CLng(FindWindow(C_VBA6_USERFORM_CLASSNAME, form.Caption))
        If formHWnd = 0 Then
            Debug.Print err.LastDllError
        End If
        ret = SetWindowPos(formHWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOSIZE)
        If ret = 0 Then
            Debug.Print err.LastDllError
        End If
    End Sub
    Public Sub MakeUserFormChildOfVBEditor(GivenFormCaption As String)
        #If VBA7 Then
            Dim VBEWindowPointer As LongPtr
            Dim UserFormWindowPointer As LongPtr
            Dim ReturnOfSetParentAPI As LongPtr
        #Else
            Dim VBEWindowPointer As Long
            Dim UserFormWindowPointer As Long
            Dim ReturnOfSetParentAPI As Long
        #End If
        Dim ErrorNumber As Long
        VBEWindowPointer = Application.VBE.MainWindow.hWnd
```

```
    UserFormWindowPointer = FindWindow(lpClassName:=USERFORM_WINDOW_CLASS, lpWindowName:= _
    GivenFormCaption)
    Const ERROR_NUMBER_FOR_SETPARENT_API = 0
    ReturnOfSetParentAPI = SetParent(hwndChild:=UserFormWindowPointer, hWndNewParent:= _
    VBEWindowPointer)
    If ReturnOfSetParentAPI = ERROR_NUMBER_FOR_SETPARENT_API Then
        ErrorNumber = err.LastDllError
        DisplayErrorText "Error With SetParent", ErrorNumber
    Else
        Debug.Print GivenFormCaption & " is child of VBE Window."
    End If
    SetForegroundWindow UserFormWindowPointer
End Sub
Sub DisplayErrorText(Context As String, ErrNum As Long)
    Dim ErrText As String
    ErrText = GetSystemErrorMessageText(ErrNum)
    MsgBox Context & vbCrLf & _
    "Error Number: " & CStr(ErrNum) & vbCrLf & _
    "Error Text:   " & ErrText, vbOKOnly
End Sub
Function GetSystemErrorMessageText(ErrorNumber As Long) As String
    Rem
    Dim ErrorText As String
    Dim ErrorTextLength As Long
    Dim FormatMessageResult As Long
    Dim LanguageID As Long
    LanguageID = 0&
    ErrorText = String$(FORMAT_MESSAGE_TEXT_LEN, " ")
    ErrorTextLength = Len(ErrorText)
    FormatMessageResult = 0&
    #If VBA7 Then
        Dim FormatMessageAPILastArgument As LongPtr
        FormatMessageAPILastArgument = 0
    #Else
        Dim FormatMessageAPILastArgument As Long
        FormatMessageAPILastArgument = 0
    #End If
    FormatMessageResult = FormatMessage( _
    dwFlags:=FORMAT_MESSAGE_FROM_SYSTEM Or FORMAT_MESSAGE_IGNORE_INSERTS, _
    lpSource:=0&, _
    dwMessageId:=ErrorNumber, _
    dwLanguageId:=0&, _
    lpBuffer:=ErrorText, _
    nSize:=ErrorTextLength, _
    Arguments:=FormatMessageAPILastArgument)
    If FormatMessageResult > 0 Then
        ErrorText = TrimToNull(ErrorText)
        GetSystemErrorMessageText = ErrorText
    Else
        GetSystemErrorMessageText = "NO ERROR DESCRIPTION AVAILABLE"
    End If
End Function
Function TrimToNull(TEXT As String) As String
    Dim NullCharIndex As Integer
    NullCharIndex = InStr(1, TEXT, vbNullChar, vbTextCompare)
    If NullCharIndex > 0 Then
        TrimToNull = Left(TEXT, NullCharIndex - 1)
    Else
        TrimToNull = TEXT
```

```
    End If
End Function
Sub AddMinimizeButtonToUserform(form As Object)
    Dim UserFormCaption As String
    UserFormCaption = form.Caption
    Dim hWnd            As Long
    Dim exLong          As Long
    hWnd = FindWindowA(vbNullString, UserFormCaption)
    exLong = GetWindowLongA(hWnd, -16)
    If (exLong And &H20000) = 0 Then
        SetWindowLongA hWnd, -16, exLong Or &H20000
    Else
    End If
End Sub
Sub UserformSetHandCursor(Optional form As Object)
    If form Is Nothing Then
        Dim Module As VBComponent
        Set Module = ActiveModule
        If Module.Type = vbext_ct_MSForm Then
            Dim ctr As MSForms.control
            For Each ctr In Module.Designer.Controls
                SetHandCursor ctr
            Next
        End If
    End If
End Sub
Sub UserformSelectedControlsSetHandCursor()
    Dim Module As VBComponent
    Set Module = ActiveModule
    If Module.Type = vbext_ct_MSForm Then
        Dim ctr As MSForms.control
        For Each ctr In SelectedControls
            SetHandCursor ctr
        Next
    End If
End Sub
Sub SetHandCursor(control As MSForms.control)
    On Error GoTo catch
    With control
        .MouseIcon = LoadPicture("C:\Users\acer\Dropbox\SOFTWARE\EXCEL\0 Alex\icons\Hand Cursor  _
        Pointer.ico")
        .MousePointer = fmMousePointerCustom
    End With
    catch:
End Sub
Sub SwitchControlNames()
    Dim ctrls As Collection
    Set ctrls = SelectedControls
    If ctrls.count <> 2 Then Exit Sub
    Dim tmp1 As String
    tmp1 = ctrls(1).Name
    Dim tmp2 As String
    tmp2 = ctrls(2).Name
    ctrls(1).Name = "tmp1"
    ctrls(2).Name = "tmp2"
    ctrls(1).Name = tmp2
    ctrls(2).Name = tmp1
End Sub
Sub SwitchControlPositions()
```

```
    Dim ctrls As Collection
    Set ctrls = SelectedControls
    If ctrls.count <> 2 Then Exit Sub
    Dim left1 As Long, left2 As Long
    Dim top1 As Long, top2 As Long
    left1 = ctrls(1).Left
    top1 = ctrls(1).Top
    left2 = ctrls(2).Left
    top2 = ctrls(2).Top
    ctrls(1).Left = left2
    ctrls(1).Top = top2
    ctrls(2).Left = left1
    ctrls(2).Top = top1
End Sub
Public Sub Reframe(form As Object, control As MSForms.control)
    Dim c As MSForms.control
    For Each c In form.Controls
        If TypeName(c) = "Frame" Then
            If Not InStr(1, c.Tag, "skip", vbTextCompare) > 0 Then
                If c.Name <> control.parent.parent.Name Then c.visible = False
            End If
        End If
    Next
    form.Controls(control.Caption).visible = True
    For Each c In form.Controls
        If TypeName(c) = "Label" Then
            If Not InStr(1, c.Tag, "skip", vbTextCompare) > 0 Then
                c.BackColor = &H534848
            End If
        End If
    Next
    control.BackColor = &H80B91E
End Sub
Sub SaveUserformOptions(form As Object, _
    Optional includeCheckbox As Boolean = True, _
    Optional includeOptionButton As Boolean = True, _
    Optional includeTextBox As Boolean = True, _
    Optional includeListbox As Boolean = True, _
    Optional includeToggleButton As Boolean = True)
    Dim ws As Worksheet
    Set ws = CreateOrSetSheet(form.Name & "_Settings", ThisWorkbook)
    ws.Cells.clear
    Dim coll As New Collection
    Dim cell As Range
    Set cell = ws.Cells(1, 1)
    Dim c As MSForms.control
    For Each c In form.Controls
        If TypeName(c) Like "CheckBox" Then
            If Not includeCheckbox Then GoTo Skip
        ElseIf TypeName(c) Like "OptionButton" Then
            If Not includeOptionButton Then GoTo Skip
        ElseIf TypeName(c) Like "TextBox" Then
            If Not includeTextBox Then GoTo Skip
        ElseIf TypeName(c) = "ListBox" Then
            If Not includeListbox Then GoTo Skip
        ElseIf TypeName(c) Like "ToggleButton" Then
            If Not includeToggleButton Then GoTo Skip
        Else
            GoTo Skip
```

```
        End If
        cell = c.Name
        Select Case TypeName(c)
        Case "TextBox", "CheckBox", "OptionButton", "ToggleButton"
            cell.OFFSET(0, 1) = c.Value
        Case "ListBox"
            Set coll = ListboxSelectedIndexes(c)
            If coll.count > 0 Then
                cell.OFFSET(0, 1) = Join(CollectionToArray(coll), ",")
            Else
                cell.OFFSET(0, 1) = -1
            End If
        End Select
        Set cell = cell.OFFSET(1, 0)
        Skip:
    Next
End Sub
Sub ListboxToRangeSelect(lBox As MSForms.ListBox)
    Dim rng As Range
    If GetInputRange(rng, "Range picker", "Select range to output listbox' list") = False Then Exit
    Sub
    rng.RESIZE(lBox.ListCount, lBox.columnCount) = CollectionsToArrayTable(ListboxSelectedValues( _
    lBox))
End Sub
Sub LoadUserformOptions(form As Object, Optional ExcludeThese As Variant)
    Dim ws As Worksheet
    Set ws = CreateOrSetSheet(form.Name & "_Settings", ThisWorkbook)
    If ws.Range("A1") = "" Then Exit Sub
    Dim cell As Range
    Set cell = ws.Cells(1, 1)
    Dim c As MSForms.control
    Dim v
    On Error Resume Next
    Do While cell <> ""
        Set c = form.Controls(cell.TEXT)
        If Not TypeName(c) = "Nothing " Then
            If Not IsInArray(cell, ExcludeThese) Then
                Select Case TypeName(c)
                Case "TextBox", "CheckBox", "OptionButton", "ToggleButton"
                    c.Value = cell.OFFSET(0, 1)
                Case "ListBox"
                    If InStr(1, cell.OFFSET(0, 1), ",") > 0 Then
                        SelectListboxItems c, Split(cell.OFFSET(0, 1), ","), True
                    Else
                        c.Selected(CInt(cell.OFFSET(0, 1))) = True
                    End If
                End Select
            End If
        End If
        Set cell = cell.OFFSET(1, 0)
    Loop
End Sub
Sub AddFormControls(controlID As String, _
    CountOrArrayOfNames As Variant, _
    Optional Captions As Variant = 0, _
    Optional Vertical As Boolean = True, _
    Optional OFFSET As Long = 0, _
    Optional form As Object)
    If IsNumeric(CountOrArrayOfNames) And IsArray(Captions) Then
```

```
        If UBound(Captions) + 1 <> CLng(CountOrArrayOfNames) Then Exit Sub
    End If
    Dim Module As VBComponent
    If form Is Nothing Then
        Set Module = ActiveModule
        If Module.Type <> vbext_ct_MSForm Then Exit Sub
    End If
    Dim c As MSForms.control
    Dim i As Long
    If IsNumeric(CountOrArrayOfNames) Then
        For i = 1 To CLng(CountOrArrayOfNames)
            If form Is Nothing Then
                Set c = Module.Designer.Controls.Add(controlID)
            Else
                Set c = form.Controls.Add(controlID)
            End If
            If Vertical Then
                c.Top = i * c.Height + i * 5 - c.Height
                c.Left = OFFSET
            Else
                c.Top = OFFSET
                c.Left = i * c.Width + i * 5 - c.Width
            End If
            If IsArray(Captions) Then
                c.Caption = Captions(i - 1)
            Else
                On Error Resume Next
                c.Caption = CountOrArrayOfNames(i - 1)
                If c.Caption = "" Then c.Caption = c.Name
                On Error GoTo 0
            End If
        Next
    Else
        For i = 1 To UBound(CountOrArrayOfNames) + 1
            If form Is Nothing Then
                Set c = Module.Designer.Controls.Add(controlID)
            Else
                Set c = form.Controls.Add(controlID)
            End If
            If Vertical Then
                c.Top = i * c.Height + i * 5 - c.Height
                c.Left = OFFSET
            Else
                c.Top = OFFSET
                c.Left = i * c.Width + i * 5 - c.Width
            End If
            c.Name = CountOrArrayOfNames(i - 1)
            If IsArray(Captions) Then
                c.Caption = Captions(i - 1)
            Else
                On Error Resume Next
                c.Caption = CountOrArrayOfNames(i - 1)
                If c.Caption = "" Then c.Caption = c.Name
                On Error GoTo 0
            End If
        Next
    End If
End Sub
Sub AddMultipleControls(ControlTypes As Variant, count As Long, Optional Vertical As Boolean = True,
```

```
Optional form As Object = Nothing)
    Dim i As Long
    For i = 1 To UBound(ControlTypes) + 1
        If Vertical Then
            AddFormControls CStr(ControlTypes(i - 1)), count, , Vertical, i * 60 - 50, form
        Else
            AddFormControls CStr(ControlTypes(i - 1)), count, , Vertical, i * 20 - 20, form
        End If
    Next
    Dim c As MSForms.control
    On Error Resume Next
    If form Is Nothing Then
        For Each c In ActiveModule.Designer.Controls
            If Not TypeName(c) Like "TextBox" Then c.AutoSize = True
        Next
    Else
        For Each c In form.Controls
            If Not TypeName(c) Like "TextBox" Then c.AutoSize = True
        Next
    End If
End Sub
Sub EditObjectProperties(obj As Variant, PropertyArguement As Variant)
    Dim i As Long
    Do While i < UBound(PropertyArguement)
        CallByName obj, PropertyArguement(i), VbLet, _
        IIf(IsNumeric(PropertyArguement(i + 1)), _
        CLng(PropertyArguement(i + 1)), _
        PropertyArguement(i + 1))
        i = i + 2
    Loop
End Sub
Sub EditObjectsProperty(obj As Collection, objProperty As String, Args As Variant)
    If obj.count <> UBound(Args) + 1 Then
        MsgBox "Not selected controls count <> arguements count"
        Exit Sub
    End If
    Dim ArgItem
    Dim i As Long
    i = obj.count
    Dim element As Variant
    For Each element In obj
        CallByName element, objProperty, VbLet, _
        IIf(IsNumeric(Args(i - 1)), _
        CLng(Args(i - 1)), _
        Args(i - 1))
        i = i - 1
    Next
End Sub
Sub RenameControlAndCode(Optional ctr As MSForms.control)
    If ctr Is Nothing Then
        If SelectedControls.count = 1 Then Set ctr = SelectedControl
        If ctr Is Nothing Then
            MsgBox "No control passed as arguement or no 1 control selected in designer"
            Exit Sub
        End If
    End If
    Dim Module As VBComponent: Set Module = ActiveModule
    If Module.Type <> vbext_ct_MSForm Then Exit Sub
    Dim OldName As String: OldName = ctr.Name
```

```
    Dim NewName As String: NewName = InputboxString
    If NewName = "" Then Exit Sub
    ctr.Name = NewName
    Dim CountOfLines As Long: CountOfLines = Module.CodeModule.CountOfLines
    If CountOfLines = 0 Then Exit Sub
    Dim strLine As String
    Dim i As Long
    For i = 1 To CountOfLines
        strLine = Module.CodeModule.Lines(i, 1)
        If InStr(1, strLine, " " & OldName & "_") > 0 Then
            If InStrExact(1, strLine, OldName & "_") > 0 Then
                Module.CodeModule.ReplaceLine (i), Replace(strLine, OldName, NewName & "_")
            End If
        End If
    Next
End Sub
Sub SortControlsHorizontally()
    SortControls False
End Sub
Sub SortControlsVertivally()
    SortControls True
End Sub
Sub SortControls(Optional SortVertically As Boolean = True)
    Dim Module As VBComponent
    Set Module = ActiveModule
    If Module.Type <> vbext_ct_MSForm Then Exit Sub
    Dim ctr As MSForms.control
    Dim coll As New Collection
    Dim lastTop As Long
    Dim lastLeft As Long
    Dim element As Variant
    For Each element In SelectedControls
        coll.Add element.Name
    Next
    Set coll = SortCollection(coll)
    lastTop = 2000
    For Each element In coll
        If Module.Designer.Controls(element).Top < lastTop Then lastTop = Module.Designer.Controls( _
        element).Top
        If Module.Designer.Controls(element).Left < lastLeft Then lastLeft = Module.Designer. _
        Controls(element).Left
    Next
    For Each element In coll
        If SortVertically = True Then
            lastTop = lastTop + Module.Designer.Controls(element).Height + 6
        Else
            lastLeft = lastLeft + Module.Designer.Controls(element).Width + 6
        End If
        Module.Designer.Controls(element).Top = lastTop
        Module.Designer.Controls(element).Left = lastLeft
    Next
End Sub
Sub CopyControlProperties(Optional control As MSForms.control)
    If control Is Nothing Then Set control = SelectedControl
    Dim ws As Worksheet: Set ws = CreateOrSetSheet("CopyControlProperties", ThisWorkbook)
    Dim PropertiesArray As Variant
    PropertiesArray = Array("Accelerator", "Alignment", "AutoSize", "AutoTab", "BackColor",  _
    "BackStyle", "BorderColor", "BorderStyle", "BoundColumn", _
    "Caption", "Children", "columnCount", "ColumnHeads", "ColumnWidths", "ControlSource",  _
```

```
    "ControlTipText", "Cycle", "DrawBuffer", "Enabled", "EnterKeyBehavior", "Expanded", _
    "FirstSibling", "FontBold", "FontSize", "ForeColor", "FullPath", "GroupName", "Height", _
    "HelpContextID", "KeepScrollBarsVisible", "LargeChange", "LastSibling", "LineStyle", "ListRows", _
    "Locked", _
    "Max", "MaxLength", "Min", "MouseIcon", "MousePointer", "MultiLine", "MultiSelect", "Next", _
    "Nodes", "Orientation", _
    "Parent", "PasswordChar", "PathSeparator", "Picture", "PictureAlignment", "PictureSizeMode", _
    "PictureTiling", "Previous", "RightToLeft", "Root", "RowSource", _
    "ScrollBars", "ScrollHeight", "ScrollLeft", "ScrollTop", "ScrollWidth", "Selected", _
    "SelectedItem", "ShowModal", "SmallChange", "Sorted", "SpecialEffect", "StartUpPosition", _
    "Style", "Tag", "Text", "TextColumn", "TripleState", "WhatsThisHelp", "Width", "Zoom")
    If ws.Range("A1") = "" Then ws.Range("A1").RESIZE(UBound(PropertiesArray) + 1) = _
    WorksheetFunction.Transpose(PropertiesArray)
    Dim PropertiesRange As Range: Set PropertiesRange = ws.Range("A1").CurrentRegion.RESIZE(, 1)
    Dim property As Range
    On Error Resume Next
    For Each property In PropertiesRange
        property.OFFSET(0, 1) = CallByName(control, property.Value, VbGet)
    Next
End Sub
Sub PasteControlProperties(Optional Controls As Collection)
    Dim control As MSForms.control
    If Controls Is Nothing Then Set Controls = SelectedControls
    Dim ws As Worksheet: Set ws = ThisWorkbook.Sheets("CopyControlProperties")
    If ws.Columns(2).SpecialCells(xlCellTypeConstants).count = 0 Then
        MsgBox "You haven't saved properties before"
        Exit Sub
    End If
    Dim PropertiesRange As Range: Set PropertiesRange = ws.Range("A1").CurrentRegion.RESIZE(, 1)
    Dim property As Range
    On Error Resume Next
    For Each control In Controls
        For Each property In PropertiesRange
            CallByName control, property.Value, VbLet, property.OFFSET(0, 1).Value
        Next
    Next
End Sub
Function SelectedControl() As MSForms.control
    Dim Module As VBComponent
    Set Module = ActiveModule
    If SelectedControls.count = 1 Then
        Dim ctl    As control
        For Each ctl In ActiveModule.Designer.Selected
            Set SelectedControl = ctl
            Exit Function
        Next ctl
    End If
End Function
Function SelectedControls() As Collection
    Dim ctl    As control
    Dim out As New Collection
    Dim Module As VBComponent
    Set Module = ActiveModule
    For Each ctl In Module.Designer.Selected
        out.Add ctl
    Next ctl
    Set SelectedControls = out
    Set out = Nothing
End Function
```

```
Sub RemoveControlsCaptions()
    Dim c As MSForms.control
    For Each c In SelectedControls
        c.Caption = ""
    Next
End Sub
Function SelectedFrameControls() As Collection
    Dim ctl    As control, c As control
    Dim out As New Collection
    Dim Module As VBComponent
    Set Module = ActiveModule
    For Each ctl In Module.Designer.Selected
        For Each c In ctl.Controls
            out.Add c
        Next
    Next ctl
    Set SelectedFrameControls = out
    Set out = Nothing
End Function
Function SelectedFrameControl() As MSForms.control
    Dim ctl    As control, c As control
    Dim out As New Collection
    Dim Module As VBComponent
    Set Module = ActiveModule
    For Each ctl In Module.Designer.Selected
        For Each c In ctl.Controls
            out.Add c
        Next
    Next ctl
    If out.count = 0 Then Exit Function
    Set SelectedFrameControl = out(1)
End Function
Sub SelectListboxItems(lBox As MSForms.ListBox, FindMe As Variant, Optional ByIndex As Boolean)
    Dim i As Long
    Select Case TypeName(FindMe)
    Case Is = "String", "Long", "Integer"
        For i = 0 To lBox.ListCount - 1
            If lBox.list(i) = CStr(FindMe) Then
                lBox.Selected(i) = True
                DoEvents
                If lBox.multiSelect = fmMultiSelectSingle Then Exit Sub
            End If
        Next
    Case Else
        Dim el As Variant
        If ByIndex Then
            For Each el In FindMe
                lBox.Selected(el) = True
            Next
        Else
            For Each el In FindMe
                For i = 0 To lBox.ListCount - 1
                    If lBox.list(i) = el Then
                        lBox.Selected(i) = True
                        DoEvents
                    End If
                Next
            Next
        End If
```

```
    End Select
End Sub
Sub CreateListboxHeader(body As MSForms.ListBox, header As MSForms.ListBox, arrHeaders)
    header.Width = body.Width
    Dim i As Long
    header.columnCount = body.columnCount
    header.ColumnWidths = body.ColumnWidths
    header.clear
    header.AddItem
    If ArrayDimensions(arrHeaders) = 1 Then
        For i = 0 To UBound(arrHeaders)
            header.list(0, i) = arrHeaders(i)
        Next i
    Else
        For i = 1 To UBound(arrHeaders, 2)
            header.list(0, i - 1) = arrHeaders(1, i)
        Next i
    End If
    body.ZOrder (1)
    header.ZOrder (0)
    header.SpecialEffect = fmSpecialEffectFlat
    header.BackColor = RGB(200, 200, 200)
    header.Height = 15
    header.Width = body.Width
    header.Left = body.Left
    header.Top = body.Top - header.Height - 1
    header.Font.Bold = True
    header.Font.Name = "Comic Sans MS"
    header.Font.Size = 9
End Sub
Sub SavePosition(form As Object)
    SaveSetting "My Settings Folder", form.Name, "Left Position", form.Left
    SaveSetting "My Settings Folder", form.Name, "Top Position", form.Top
End Sub
Sub LoadPosition(form As Object)
    If GetSetting("My Settings Folder", form.Name, "Left Position") = "" _
        And GetSetting("My Settings Folder", form.Name, "Top Position") = "" Then
        form.StartUpPosition = 1
    Else
        form.Left = GetSetting("My Settings Folder", form.Name, "Left Position")
        form.Top = GetSetting("My Settings Folder", form.Name, "Top Position")
    End If
End Sub
Rem
Sub ResizeUserformToFitControls(form As Object)
    form.Width = 0
    form.Height = 0
    Dim ctr As MSForms.control
    Dim myWidth
    myWidth = form.InsideWidth
    Dim myHeight
    myHeight = form.InsideHeight
    For Each ctr In form.Controls
        If ctr.visible = True Then
            If ctr.Left + ctr.Width > myWidth Then myWidth = ctr.Left + ctr.Width
            If ctr.Top + ctr.Height > myHeight Then myHeight = ctr.Top + ctr.Height
        End If
    Next
    form.Width = myWidth + form.Width - form.InsideWidth + 10
```

```
    form.Height = myHeight + form.Height - form.InsideHeight + 10
End Sub
Function whichOption(Frame As Variant, controlType As String) As Variant
    Dim subControl As MSForms.control
    Dim out As New Collection
    Dim control As MSForms.control
    For Each control In Frame.Controls
        If UCase(TypeName(control)) = UCase("Frame") Then
            If UCase(TypeName(control)) = UCase(controlType) Then
                If control.Value = True Then
                    out.Add control
                End If
            End If
        End If
    Next
    If out.count = 1 Then
        whichOption = out(1)
    ElseIf out.count > 1 Then
        Set whichOption = out
    End If
End Function
Public Sub flashControl(ctr As MSForms.control, blinkCount As Integer)
    Dim lngTime As Long
    Dim i As Integer
    If blinkCount Mod 2 <> 0 Then blinkCount = blinkCount + 1
    For i = 1 To blinkCount * 2
        lngTime = getTickCount
        If ctr.visible = True Then
            ctr.visible = False
        Else
            ctr.visible = True
        End If
        DoEvents
        Do While getTickCount - lngTime < 200
        Loop
    Next
End Sub
Public Function TextOfControl(c As control) As Variant
    Dim out As New Collection
    If TypeName(c) = "TextBox" Then
        If c.SelLength = 0 Then
            TextOfControl = c.TEXT
        Else
            TextOfControl = c.SelText
        End If
    ElseIf TypeName(c) = "ComboBox" Then
        If c.Style < 2 Then
            TextOfControl = c.TEXT
        Else
            TextOfControl = ""
        End If
    ElseIf TypeName(c) = "ListBox" Then
        Set out = ListboxSelectedValues(c)
        If out.count > 0 Then
            TextOfControl = CollectionToArray(out)
        Else
            TextOfControl = ""
        End If
    End If
```

```
End Function
Public Function ListboxContains(lBox As MSForms.ListBox, str As String, _
    Optional ColumnIndexZeroBased As Long = -1, _
    Optional CaseSensitive As Boolean = False) As Boolean
    Dim i      As Long
    Dim n      As Long
    Dim sTemp  As String
    If ColumnIndexZeroBased > lBox.columnCount - 1 Or ColumnIndexZeroBased < 0 Then
        ColumnIndexZeroBased = -1
    End If
    n = lBox.ListCount
    If ColumnIndexZeroBased <> -1 Then
        For i = n - 1 To 0 Step -1
            If CaseSensitive = True Then
                sTemp = lBox.list(i, ColumnIndexZeroBased)
            Else
                str = LCase(str)
                sTemp = LCase(lBox.list(i, ColumnIndexZeroBased))
            End If
            If InStr(1, sTemp, str) > 0 Then
                ListboxContains = True
                Exit Function
            End If
        Next i
    Else
        Dim columnCount As Long
        n = lBox.ListCount
        For i = n - 1 To 0 Step -1
            For columnCount = 0 To lBox.columnCount - 1
                If CaseSensitive = True Then
                    sTemp = lBox.list(i, columnCount)
                Else
                    str = LCase(str)
                    sTemp = LCase(lBox.list(i, columnCount))
                End If
                If InStr(1, sTemp, str) > 0 Then
                    ListboxContains = True
                    Exit Function
                End If
            Next columnCount
        Next i
    End If
End Function
Public Sub FilterListboxByColumn(lBox As MSForms.ListBox, str As String, _
    Optional ColumnIndexZeroBased As Long = -1, Optional CaseSensitive As Boolean = False)
    Dim i              As Long
    Dim n              As Long
    Dim sTemp          As String
    If ColumnIndexZeroBased > lBox.columnCount - 1 Or ColumnIndexZeroBased < 0 Then
        ColumnIndexZeroBased = -1
    End If
    n = lBox.ListCount
    If ColumnIndexZeroBased <> -1 Then
        For i = n - 1 To 0 Step -1
            If CaseSensitive = True Then
                sTemp = lBox.list(i, ColumnIndexZeroBased)
            Else
                str = LCase(str)
                sTemp = LCase(lBox.list(i, ColumnIndexZeroBased))
```

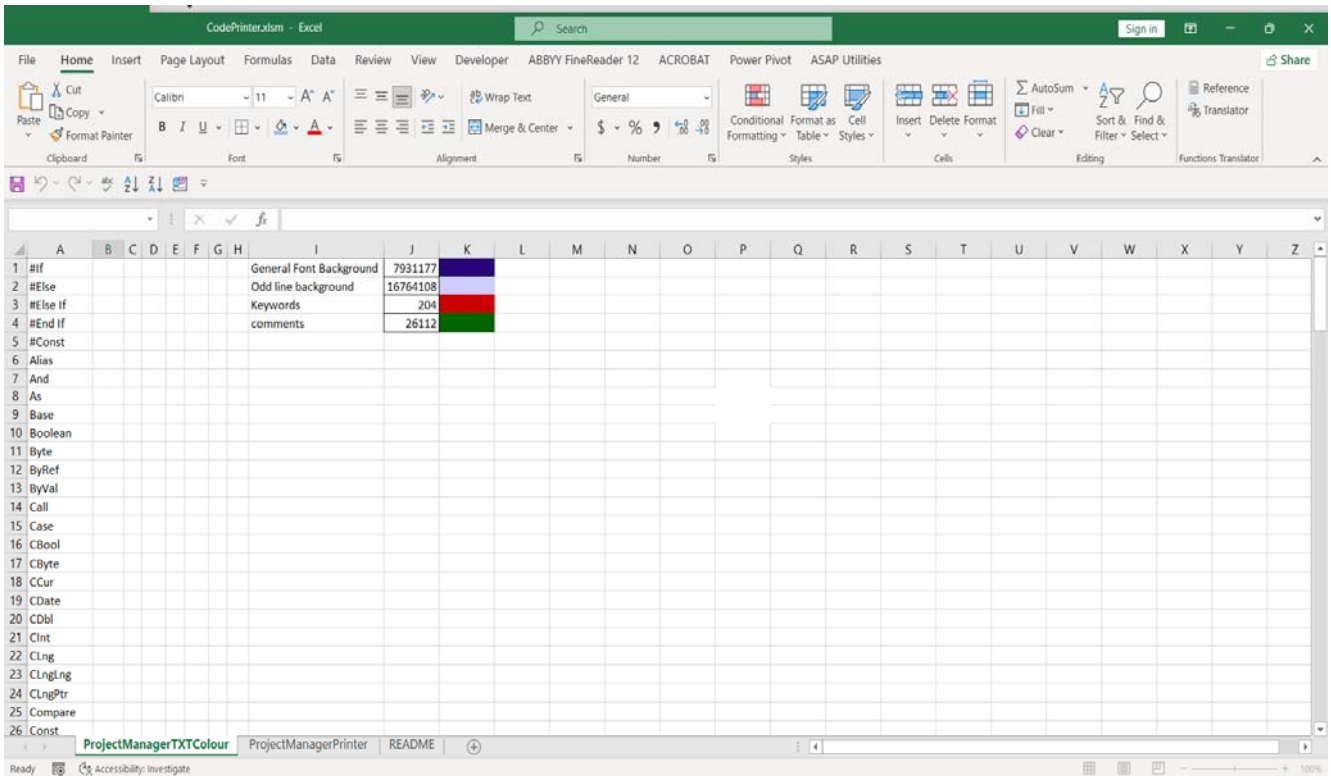
```
            End If
            If InStr(1, sTemp, str) = 0 Then
                lBox.RemoveItem (i)
            End If
        Next i
    Else
        Dim columnCount As Long
        n = lBox.ListCount
        For i = n - 1 To 0 Step -1
            For columnCount = 0 To lBox.columnCount - 1
                If CaseSensitive = True Then
                    sTemp = lBox.list(i, columnCount)
                Else
                    str = LCase(str)
                    sTemp = LCase(lBox.list(i, columnCount))
                End If
                If InStr(1, sTemp, str) > 0 Then
                Else
                    If columnCount = lBox.columnCount - 1 Then
                        lBox.RemoveItem (i)
                    End If
                End If
            Next columnCount
        Next i
    End If
End Sub
Public Sub SortListboxOnColumn(lBox As MSForms.ListBox, OnColumn As Long)
    Dim vntData As Variant
    Dim vntTempItem As Variant
    Dim lngOuterIndex As Long
    Dim lngInnerIndex As Long
    Dim lngSubItemIndex As Long
    vntData = lBox.list
    For lngOuterIndex = LBound(vntData, 1) To UBound(vntData, 1) - 1
        For lngInnerIndex = lngOuterIndex + 1 To UBound(vntData, 1)
            If vntData(lngOuterIndex, OnColumn) > vntData(lngInnerIndex, OnColumn) Then
                For lngSubItemIndex = 0 To lBox.columnCount - 1
                    vntTempItem = vntData(lngOuterIndex, lngSubItemIndex)
                    vntData(lngOuterIndex, lngSubItemIndex) = vntData(lngInnerIndex,  _
                    lngSubItemIndex)
                    vntData(lngInnerIndex, lngSubItemIndex) = vntTempItem
                Next
            End If
        Next lngInnerIndex
    Next lngOuterIndex
    lBox.clear
    lBox.list = vntData
End Sub
Function ListboxSelectedIndexes(lBox As MSForms.ListBox) As Collection
    Dim i As Long
    Dim SelectedIndexes As Collection
    Set SelectedIndexes = New Collection
    If lBox.ListCount > 0 Then
        For i = 0 To lBox.ListCount - 1
            If lBox.Selected(i) Then SelectedIndexes.Add i
        Next i
    End If
    Set ListboxSelectedIndexes = SelectedIndexes
End Function
```

```
Function ListboxSelectedValues(listboxCollection As Variant) As Collection
    Dim i As Long
    Dim listItem As Long
    Dim selectedCollection As Collection
    Set selectedCollection = New Collection
    Dim listboxCount As Long
    If TypeName(listboxCollection) = "Collection" Then
        For listboxCount = 1 To listboxCollection.count
            If listboxCollection(listboxCount).ListCount > 0 Then
                For listItem = 0 To listboxCollection(listboxCount).ListCount - 1
                    If listboxCollection(listboxCount).Selected(listItem) Then
                        selectedCollection.Add CStr(listboxCollection(listboxCount).list(listItem, _
                        listboxCollection(listboxCount).BoundColumn - 1))
                    End If
                Next listItem
            End If
        Next listboxCount
    Else
        If listboxCollection.ListCount > 0 Then
            For i = 0 To listboxCollection.ListCount - 1
                If listboxCollection.Selected(i) Then
                    selectedCollection.Add listboxCollection.list(i, listboxCollection.BoundColumn _
                    - 1)
                End If
            Next i
        End If
    End If
    Set ListboxSelectedValues = selectedCollection
End Function
Function ListboxSelectedCount(listboxCollection As Variant) As Long
    Dim i As Long
    Dim listItem As Long
    Dim selectedCollection As Collection
    Set selectedCollection = New Collection
    Dim listboxCount As Long
    Dim SelectedCount As Long
    If TypeName(listboxCollection) = "Collection" Then
        For listboxCount = 1 To listboxCollection.count
            If listboxCollection(listboxCount).ListCount > 0 Then
                For listItem = 0 To listboxCollection(listboxCount).ListCount - 1
                    If listboxCollection(listboxCount).Selected(listItem) = True Then
                        SelectedCount = SelectedCount + 1
                    End If
                Next listItem
            End If
        Next listboxCount
    Else
        If listboxCollection.ListCount > 0 Then
            For i = 0 To listboxCollection.ListCount - 1
                If listboxCollection.Selected(i) = True Then
                    SelectedCount = SelectedCount + 1
                End If
            Next i
        End If
    End If
    ListboxSelectedCount = SelectedCount
End Function
Public Sub ShowUserform(FormName As String)
    Dim frm As Object
```

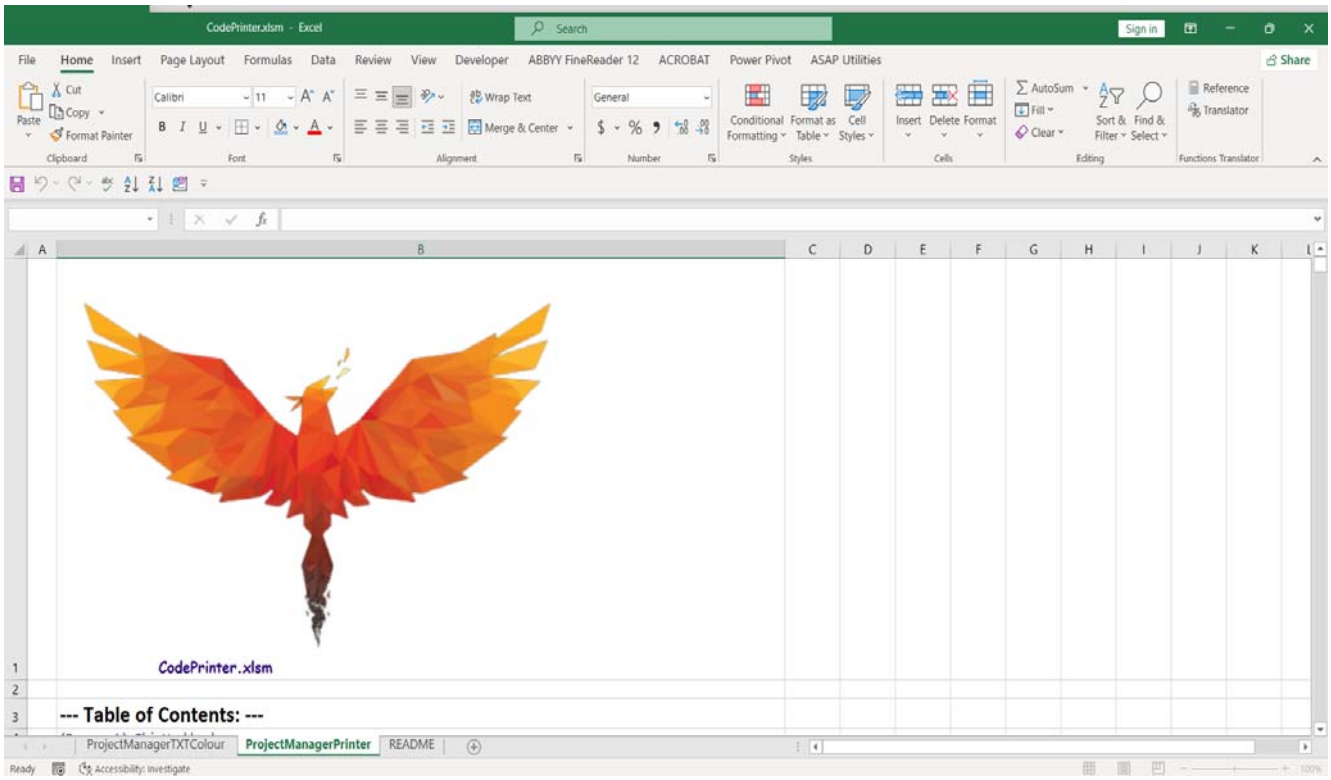
```
    If IsLoaded(FormName) = True Then
        For Each frm In VBA.UserForms
            If frm.Name = FormName Then
                frm.Show
                Exit Sub
            End If
        Next frm
    Else
        Dim oUserForm As Object
        On Error GoTo err
        Set oUserForm = UserForms.Add(FormName)
        oUserForm.Show (vbModeless)
        Exit Sub
    End If
    err:
    Select Case err.Number
    Case 424:
        MsgBox "The Userform with the name " & FormName & " was not found.", vbExclamation, "Load _
        userforn by name"
    Case Else:
        MsgBox err.Number & ": " & err.Description, vbCritical, "Load userforn by name"
    End Select
End Sub
Sub ResizeControlColumns(ListboxOrCombobox As MSForms.control, Optional ResizeControl As Boolean, _
Optional ResizeListbox As Boolean)
    If ListboxOrCombobox.ListCount = 0 Then Exit Sub
    Application.ScreenUpdating = False
    Dim ws As Worksheet
    Set ws = CreateOrSetSheet("ListboxColumnwidth", ThisWorkbook)
    Dim rng As Range
    Set rng = ws.Range("A1")
    Set rng = rng.RESIZE(UBound(ListboxOrCombobox.list) + 1, ListboxOrCombobox.columnCount)
    rng = ListboxOrCombobox.list
    rng.Font.Name = ListboxOrCombobox.Font.Name
    rng.Font.Size = ListboxOrCombobox.Font.Size + 2
    rng.Columns.AutoFit
    Dim sWidth As String
    Dim vR() As Variant
    Dim n As Integer
    Dim cell As Range
    For Each cell In rng.RESIZE(1)
        n = n + 1
        ReDim Preserve vR(1 To n)
        vR(n) = cell.EntireColumn.Width
    Next cell
    sWidth = Join(vR, ";")
    With ListboxOrCombobox
        .ColumnWidths = sWidth
        .BorderStyle = fmBorderStyleSingle
    End With
    Application.DisplayAlerts = False
    ws.Delete
    Application.DisplayAlerts = True
    Application.ScreenUpdating = True
    If ResizeListbox = False Then Exit Sub
    Dim w As Long
    Dim i As Long
    For i = LBound(vR) To UBound(vR)
        w = w + vR(i)
```

```
    Next
    DoEvents
    ListboxOrCombobox.Width = w + 10
End Sub
Sub DeselectListbox(lBox As MSForms.ListBox)
    If lBox.ListCount <> 0 Then
        Dim i As Long
        For i = 0 To lBox.ListCount - 1
            lBox.Selected(i) = False
        Next i
    End If
End Sub
Public Sub SelectDeselectAll(lBox As MSForms.ListBox, Optional toSelect As Boolean)
    If lBox.ListCount = 0 Then Exit Sub
    Dim i As Long
    For i = 0 To lBox.ListCount - 1
        lBox.Selected(i) = toSelect
    Next
End Sub
Sub SelectControItemsByFilter(lBox As MSForms.ListBox, criteria As String)
    SelectDeselectAll lBox
    If criteria = "" Then Exit Sub
    Dim i As Long
    For i = 0 To lBox.ListCount - 1
        If UCase(lBox.list(i, 1)) Like "*" & UCase(criteria) & "*" Then
            lBox.Selected(i) = True
        End If
    Next i
End Sub
```

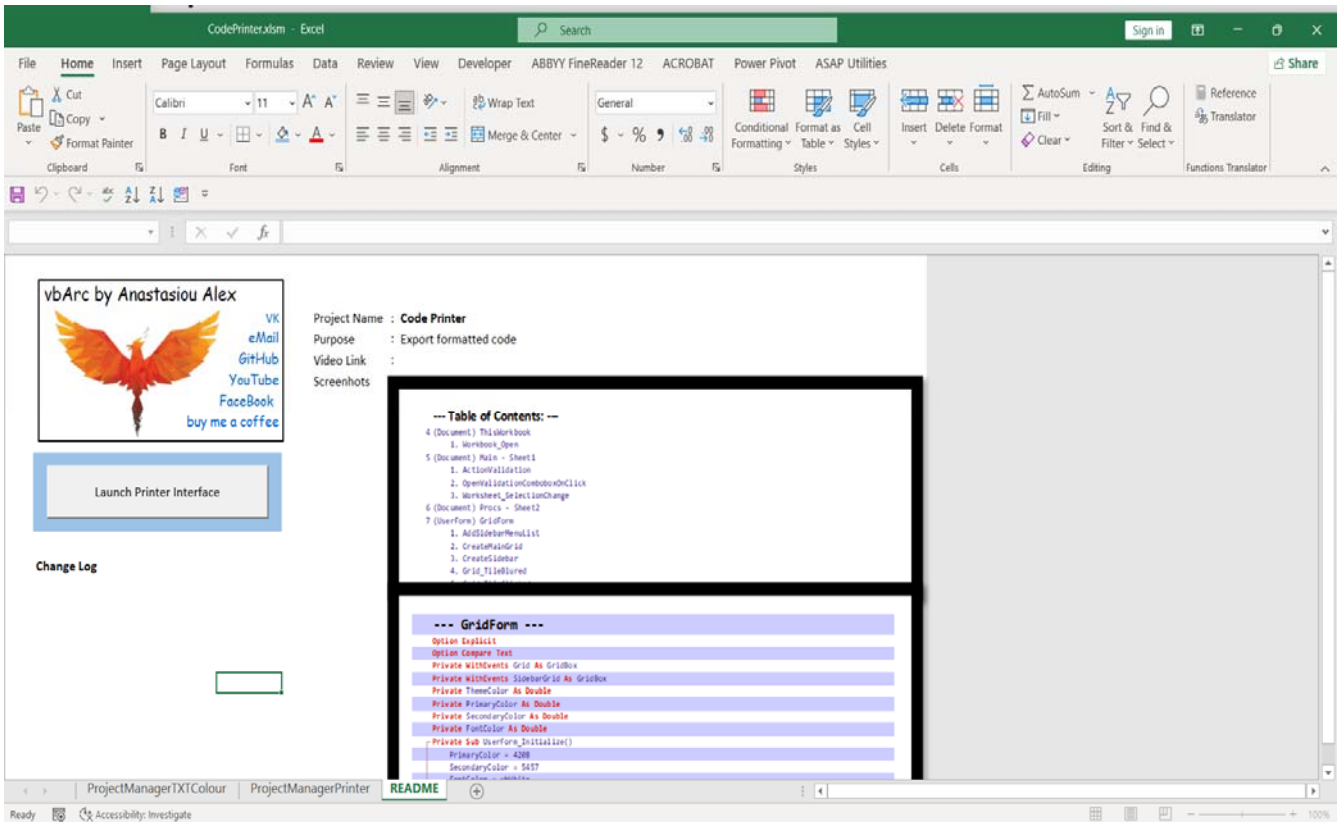
--- Image of Worksheet : ProjectManagerTXTColour ---



--- Image of Worksheet : ProjectManagerPrinter ---



--- Image of Worksheet : README ---



--- Image of Userform : uCodePrinter ---