```vb
Imports SldWorks
Imports SwCommands
Imports SwConst
Imports System.IO
Imports System.IO.Compression
Imports System.IO.Compression.ZipFile
Imports Microsoft.Office.Interop.Excel

Public Class SWFunctions

    Public Shared swApp As SldWorks.SldWorks


    Public Shared CusProperties_Part As CustomPropertyManager
    Public Shared CusProperties_Assy As CustomPropertyManager

    Public Shared swModelDocExt_Part As ModelDocExtension
    Public Shared swModelDocExt_Assy As ModelDocExtension

    Public Shared Add_BOM As Boolean = False
    Public Shared swAssy_Docs As New List(Of Assy_Docs)
    Public Shared swPart_Docs As New List(Of Part_Docs)
    Public Shared swDwg_Docs As New List(Of Drawing_Docs)
    Shared swComp_Assy As String


    Class Assy_Docs
        Public Comp As String
        Public subcomp As String
        Public instance_ID As String
        Public Part_Number As String = "Null"
        Public Nomenclature As String = "Null"
        Public Spec As String = "Null"
        Public Description As String = "Null"
        Public Material As String = "Null"
        Public Weight As String = "Null"
        Public Name As String
        Public Counter As Integer
        Public Used As Boolean

        Public Sub New(s1 As String, s2 As String, s3 As String, s4 As String, 
          s5 As String, s6 As String, s7 As String, s8 As String, s9 As String)
            Comp = s1
            subcomp = s2
            instance_ID = s3
            Part_Number = s4
            Nomenclature = s5
            Description = s6
            Spec = s7
            Material = s8
            Weight = s9

        End Sub

    End Class

    Class Part_Docs
        Public Comp As String
        Public subcomp As String
        Public instance_ID As String
        Public Part_Number As String = "Null"
        Public Nomenclature As String = "Null"
        Public Spec As String = "Null"
        Public Description As String = "Null"
        Public Material As String = "Null"
        Public Weight As String = "Null"
        Public Name As String
        Public Counter As Integer
        Public Used As Boolean

        Public Sub New(s1 As String, s2 As String, s3 As String, s4 As String,
          s5 As String, s6 As String, s7 As String, s8 As String, s9 As String)
            Comp = s1
            subcomp = s2
            instance_ID = s3
            Part_Number = s4
            Nomenclature = s5
            Description = s6
            Spec = s7
            Material = s8
            Weight = s9

        End Sub

    End Class

    Class Drawing_Docs
        Public Comp As String
        Public subcomp As String
        Public instance_ID As String
        Public Part_Number As String = "Null"
        Public Nomenclature As String = "Null"
        Public Spec As String = "Null"
        Public Description As String = "Null"
        Public Material As String = "Null"
        Public Name As String
        Public Counter As Integer
        Public Used As Boolean

        Public Sub New(s1 As String, s2 As String, s3 As String, s4 As String,
          s5 As String, s6 As String, s7 As String, s8 As String)
            Comp = s1
            subcomp = s2
            instance_ID = s3
            Part_Number = s4
            Nomenclature = s5
            Spec = s6
            Description = s7
            Material = s8

        End Sub

    End Class


    Shared Sub Connect_SW()
        Dim SWProcess() As Process
        Dim SWProcess2() As Process

        Dim app1 As Object = Nothing
        Dim SW_Path = String.Empty
        Dim aProcess As System.Diagnostics.Process
        If swApp Is Nothing Then


            SWProcess = Process.GetProcessesByName("SLDWORKS")

            'MsgBox(SWProcess.Count) '& " --- " & SWProcess2.Count)
            If SWProcess.Count > 1 Then
                For k = 0 To SWProcess.Count - 1

                    'SWProcess(k).CloseMainWindow()
                    aProcess = System.Diagnostics.Process.GetProcessById
                    (SWProcess(k).Id)
                    aProcess.Kill() 'kills all .exe
                    'aProcess.Close()   'doesn't work at all
                    'aProcess.CloseMainWindow() 'closes windows that are opened
                    and visible

                    Threading.Thread.Sleep(500)
                Next

            End If
            SWProcess = Process.GetProcessesByName("SLDWORKS")
            If SWProcess.Count = 1 Then
                'app1 = TryCast(CreateObject("SldWorks.Application"),
                  SolidWorks.Interop.sldworks.ISldWorks)
                app1 = CreateObject("SldWorks.Application")
                Threading.Thread.Sleep(1000)
            Else
                For i = 10 To 0 Step -1
                    If i = 0 Then
                        SW_Path = "C:\Program Files\SOLIDWORKS Corp\SOLIDWORKS
                    \SLDWORKS.exe"
                        Exit For
                    End If
```

```vb
                    SW_Path = "C:\Program Files\SOLIDWORKS Corp\SOLIDWORKS (" & i & ")"
                    'MsgBox(SW_Path & " ---- " & Directory.Exists(SW_Path))
                    If Directory.Exists(SW_Path) Then
                        SW_Path = SW_Path & "\SLDWORKS.exe"
                        Exit For
                    End If

                Next
                Process.Start(SW_Path)
                SWProcess2 = Process.GetProcessesByName("SLDWORKS")
                Threading.Thread.Sleep(1000)
                'MsgBox(SWProcess2.Count)
                If SWProcess2.Count = 1 Then
                    'app1 = TryCast(CreateObject("SldWorks.Application"), SolidWorks.Interop.sldworks.ISldWorks)
                    app1 = CreateObject("SldWorks.Application")
                    'MsgBox("1")
                End If
                '
            End If


            app1.Visible = True
            app1.FrameState = 1
            swApp = app1
        End If


    End Sub

    Shared Sub Remove_SW()
        If swApp IsNot Nothing Then

            Threading.Thread.Sleep(250)
            System.Runtime.InteropServices.Marshal.ReleaseComObject(swApp)
            swApp = Nothing

        End If

    End Sub


    Shared Function Opened_Docs()
            Dim swApp As SldWorks.SldWorks
            Dim Open_Docs As Object
            Dim count As Integer

            swApp = CreateObject("SldWorks.Application")

            count = swApp.GetDocumentCount
            Open_Docs = swApp.GetDocuments

            Return Open_Docs
        End Function


    Shared Function Sheet_Rename()

            Dim swApp As SldWorks.SldWorks
            Dim swDoc As ModelDoc2
            Dim swDraw As DrawingDoc
            Dim SWSheet As Sheet
            Dim swModelDocExt As ModelDocExtension

            swApp = CreateObject("SldWorks.Application")
            swDoc = swApp.ActiveDoc()

            If swDoc Is Nothing Then
                Functions.Error_Form()
            Else

                If swDoc.GetType <> 3 Then
                    Functions.Error_Form(, "this is not a drawing file")
                Else
                    swDraw = swDoc
                    If swDraw Is Nothing Then
                        Functions.Error_Form(, "No Drawing sheet loaded")
                    Else
                        Dim sheetnames() As String
                        Dim Sheetnumbers As Integer
                        Start_Window.SW_Doc.Text = "Sheet Renaming in Proocess"

                        swApp.Visible = True
                        Sheetnumbers = swDraw.GetSheetCount
                        sheetnames = swDraw.GetSheetNames

                        For i = 1 To (Sheetnumbers)
                            swDraw.ActivateSheet(sheetnames(i - 1))
                            SWSheet = swDraw.GetCurrentSheet
                            SWSheet.SetName("Renaming" & i)
                        Next

                        sheetnames = swDraw.GetSheetNames
                        For i = 1 To (Sheetnumbers)
                            swDraw.ActivateSheet(sheetnames(i - 1))
                            SWSheet = swDraw.GetCurrentSheet
                            SWSheet.SetName("Sheet" & i)
                            swModelDocExt = swDoc.Extension
                            swModelDocExt.ViewZoomToSheet()
                        Next

                    End If
                End If
                Return True
            End If

            Return False
        End Function


    Shared Function PDF()

        Dim swApp As SldWorks.SldWorks
        Dim swDoc As ModelDoc2

        Dim CusProperties As CustomPropertyManager

        Dim bool As Boolean

        swApp = CreateObject("SldWorks.Application")
        swDoc = swApp.ActiveDoc

        If swDoc Is Nothing Then
            Functions.Error_Form()
        Else
            If swDoc.GetType = 3 Then

                Dim Title = String.Empty
                Dim REV = String.Empty
                Dim REV_1 = String.Empty
                Dim Valout = String.Empty
                Dim Res_Valout = String.Empty
                Dim Resolved As Boolean
                Dim Cus_Prop_val As Integer
                Dim FileName = String.Empty
                Dim PDF_Path = String.Empty
                Dim PathName = String.Empty
                Dim Structural = String.Empty
                Dim Drawings_Major = String.Empty
                Dim Drawings_Minor = String.Empty
                Dim Plane_Folder = String.Empty
                Dim Released_to_Inspection = "(4) Released to Inspection"
                Dim Released_As = String.Empty

                Dim swLayer As Layer
                Dim swLayerMgr = swDoc.GetLayerManager

                PathName = swDoc.GetPathName()
                Debug.Print(PathName)

                FileName = System.IO.Path.GetFileNameWithoutExtension(PathName)
                Debug.Print(FileName)

                PathName = System.IO.Path.GetDirectoryName(PathName)
                Debug.Print(PathName)
```

```

                    swDoc.ClearSelection2(True)

                    Structural = Directory.GetParent(PathName).FullName
                    Debug.Print(Structural)

                    Drawings_Major = Directory.GetParent(Structural).FullName
                    Debug.Print(Drawings_Major)

                    Dim length = Drawings_Major.Length - 18
                    Debug.Print(length)

                    Dim Major = Drawings_Major.Substring(length, 3)
                    Debug.Print(Major)

                    Plane_Folder = Directory.GetParent(Drawings_Major).FullName
                    Debug.Print(Plane_Folder)

                    'If Major = "(2)" Then
                    '    'Drawing is a major
                    '    Released_As = "Major"
                    '    PDF_Path = Path.Combine(Plane_Folder, Released_to_Inspection)
                    '    PDF_Path = Path.Combine(PDF_Path, Released_As)

                    '    System.IO.Directory.CreateDirectory(PDF_Path)

                    'ElseIf Major = "(3)" Then

                    '    'Drawing is a minor
                    '    Released_As = "Minor"

                    '    PDF_Path = Path.Combine(Drawings_Major, Released_to_Inspection)
                    '    PDF_Path = Path.Combine(PDF_Path, Released_As)

                    '    System.IO.Directory.CreateDirectory(PDF_Path)

                    'Else
                    Released_As = "PDF"
                    PDF_Path = Path.Combine(PathName, "PDF RELEASES")

                    'End If

                    CusProperties = swDoc.Extension.CustomPropertyManager("")

                    Title = CusProperties.Get("TITLE")
                    REV_1 = CusProperties.Get("REV 1")

                    If REV_1.StartsWith("P") Then
                        REV = CusProperties.Get("REV 1")
                    Else
```

```
                REV = CusProperties.Get("CURRENT REV")
            End If

            Try
                Cus_Prop_val = CusProperties.Get6("CURRENT REV", False, Valout, REV, Resolved, False)

                Select Case REV
                    Case "P1"
                        PDF_Path = Path.Combine(PDF_Path, "Preliminaries")

                    Case "P2"
                        PDF_Path = Path.Combine(PDF_Path, "Preliminaries")
                    Case "P3"
                        PDF_Path = Path.Combine(PDF_Path, "Preliminaries")
                    Case "P4"
                        PDF_Path = Path.Combine(PDF_Path, "Preliminaries")
                    Case "P5"
                        PDF_Path = Path.Combine(PDF_Path, "Preliminaries")
                    Case "P6"
                        PDF_Path = Path.Combine(PDF_Path, "Preliminaries")
                    Case "P7"
                        PDF_Path = Path.Combine(PDF_Path, "Preliminaries")
                    Case "I/R"
                        PDF_Path = Path.Combine(PDF_Path, "IR")
                        REV = "IR"
                        swLayer = swLayerMgr.GetLayer("SIGBLK")
                        swLayer.Visible = False
                    Case "A"
                        PDF_Path = Path.Combine(PDF_Path, "A")
                        swLayer = swLayerMgr.GetLayer("SIGBLK")
                        swLayer.Visible = False
                    Case "B"
                        PDF_Path = Path.Combine(PDF_Path, "B")
                        swLayer = swLayerMgr.GetLayer("SIGBLK")
                        swLayer.Visible = False
                    Case "C"
                        PDF_Path = Path.Combine(PDF_Path, "C")
                        swLayer = swLayerMgr.GetLayer("SIGBLK")
                        swLayer.Visible = False
                    Case "D"
                        PDF_Path = Path.Combine(PDF_Path, "D")
                        swLayer = swLayerMgr.GetLayer("SIGBLK")
                        swLayer.Visible = False
                    Case Else
                        REV = ""
                End Select

            Catch 'nl As System.NullReferenceException
                Functions.Error_Form("MISSING LAYER", "SIGBLK")

            End Try
```

```

                System.IO.Directory.CreateDirectory(PDF_Path)

                'If REV = "I/R" Then
                '   REV = "IR"
                '   swLayer = swLayerMgr.GetLayer("SIGBLK")
                '   swLayer.Visible = False
                'End If

                PDF_Path = Path.Combine(PDF_Path, FileName)
                Title = PDF_Path + " " + REV + " (" + Title + ")"

                bool = swDoc.SaveAs4(Title + ".PDF",
                  swSaveAsVersion_e.swSaveAsCurrentVersion,
                  swSaveAsOptions_e.swSaveAsOptions_Silent, 0, 0)
                If bool = False Then
                    Functions.Error_Form("PDF Error", "PDF did not save",,,,
                    False,)
                End If

            Else
                Functions.Error_Form("Document not a SLDDRW", "No Drawing File
                  opened")

                End If
                Return True
            End If

            Return False
        End Function


    Shared Function SW_FileName(path As String)
            Dim FileName As String = path

            FileName = FileName.Remove(0, FileName.LastIndexOf("\") + 1)
            FileName = FileName.Remove(FileName.LastIndexOf("."))

            Return FileName
        End Function


    Shared Function SW_Extension(path As String)
            Dim Extension As String = path

            Extension = Extension.Remove(0, Extension.LastIndexOf("."))

            Return Extension
        End Function


    Shared Function PackandGo()
```

```

        Dim swApp As SldWorks.SldWorks
        Dim swDoc As ModelDoc2
        Dim swPackAndGo As PackAndGo

        Dim status As Boolean
        Dim pgFileNames As Object = Nothing
        Dim pgFileStatus As Object = Nothing
        Dim statuses As Object
        Dim Open_Docs As Object
        Dim Filename = String.Empty
        Dim Pathname = String.Empty
        Dim ASSYname = String.Empty
        Dim MyFolder = String.Empty
        Dim ExistingFile = String.Empty

        Dim TimeNow As DateTime = DateTime.Now
        Dim format As String = "MM-dd-yyyy HH-mm-ss"

        swApp = CreateObject("SldWorks.Application")
        swApp.Visible = True
        swDoc = swApp.ActiveDoc

        'If swDoc Is Nothing Then
        If swDoc.GetType <> 3 Then

            Functions.Error_Form()
        Else

            swDoc.ForceRebuild3(True)
            swDoc.Save3(1, 1, 2)

            'check for existing file is true
            ExistingFile = swDoc.GetPathName

            If ExistingFile = "" Then

                Functions.Error_Form("File Error", "File is not Saved",, "Please save before using Pack n Go",,,)

            Else

                Filename = System.IO.Path.GetFileNameWithoutExtension(ExistingFile)
                Pathname = System.IO.Path.GetDirectoryName(ExistingFile)
                ASSYname = Pathname + "\" + Filename + " (Pack and Go)"


                swPackAndGo = swDoc.Extension.GetPackAndGo
                swPackAndGo.IncludeDrawings = True
                swPackAndGo.IncludeSimulationResults = False
                'swPackAndGo.IncludeSuppressed = True
```

```vb
                swPackAndGo.IncludeToolboxComponents = True
                swPackAndGo.FlattenToSingleFolder = True


                status = swPackAndGo.GetDocumentSaveToNames(pgFileNames, pgFileStatus)

                Dim External_Files As New List(Of String)
                Dim exclude = {".slddrw", ".sldasm", ".sldprt"}
                Dim Extension As String

                For Each myFile As String In Directory.GetFiles(Pathname)
                    Extension = LCase(Path.GetExtension(myFile))
                    If Extension = exclude(0) Or Extension = exclude(1) Or Extension = exclude(2) Then
                        'MsgBox("failed")
                    Else
                        External_Files.Add(myFile)
                        MsgBox(myFile)
                    End If

                Next

                'Dim external_Files = Directory.GetFiles(Pathname)

                status = swPackAndGo.AddExternalDocuments(External_Files.ToArray)

                'status = swPackAndGo.AddExternalDocuments(swPackAndGo.GetDocumentNames(pgFileNames)) 'get files in directory as an array


                'Puts Pack and Go files in native Directory
                status = swPackAndGo.SetSaveToName(True, Pathname)

                statuses = swDoc.Extension.SavePackAndGo(swPackAndGo)

                'Puts Pack and Go files in native Directory
                status = swPackAndGo.SetSaveToName(True, ASSYname)

                swPackAndGo.AddPrefix = Filename + " - "

                'swPackAndGo.

                'Puts Pack and Go files in new Directory named "Assembly name (Pack and Go)"
                'status = swPackAndGo.SetSaveToName(True, ASSYname)

                statuses = swDoc.Extension.SavePackAndGo(swPackAndGo)

                Open_Docs = Opened_Docs()

                If swDoc.GetType = CInt(3) Then

                    For i = LBound(Open_Docs) To UBound(Open_Docs)
                        swDoc = Open_Docs(i)
                        Dim F_Name As String
                        Dim errors As Integer
                        Dim MoreErr As Integer

                        If swDoc.GetType = 2 Or swDoc.GetType = 3 Then
                            F_Name = SWFunctions.SW_FileName(swDoc.GetPathName())
                            swApp.IActivateDoc3(F_Name, False, errors)
                            swDoc.Save3(1, errors, MoreErr)
                            swApp.QuitDoc(F_Name)

                        End If

                    Next

                    swApp.QuitDoc(ExistingFile)
                    swApp.OpenDoc6(ExistingFile, 3, 1, "", 1, 2)

                ElseIf swDoc.GetType = CInt(2) Then

                    For i = LBound(Open_Docs) To UBound(Open_Docs)
                        swDoc = Open_Docs(i)
                        Dim F_Name As String
                        Dim errors As Integer
                        Dim MoreErr As Integer


                        If swDoc.GetType = 2 Or swDoc.GetType = 3 Then
                            F_Name = SWFunctions.SW_FileName(swDoc.GetPathName())
                            swApp.IActivateDoc3(F_Name, False, errors)
                            swDoc.Save3(1, errors, MoreErr)

                            swApp.QuitDoc(F_Name)

                            i = UBound(Open_Docs)

                        End If

                    Next

                    MsgBox(ExistingFile)
                    swApp.CloseAllDocuments(True)
                    swApp.OpenDoc6(ExistingFile, 2, 1, "", 1, 2)

                Else
                    Functions.Error_Form("Pack And Go Failure", "Incompatible
                    file type to use Pack and Go feature",,,,,)
                End If

                If Directory.Exists(ASSYname) Then

                    status = swPackAndGo.SetSaveToName(True, ASSYname + " - " + 
                    TimeNow.ToString(Format))
                    statuses = swDoc.Extension.SavePackAndGo(swPackAndGo)
                Else
                    status = swPackAndGo.SetSaveToName(True, ASSYname)
                    statuses = swDoc.Extension.SavePackAndGo(swPackAndGo)
                End If


            End If
            If File.Exists(ASSYname + ".zip") Then

                System.IO.Compression.ZipFile.CreateFromDirectory(ASSYname, 
                  ASSYname + " - " + TimeNow.ToString(Format) + ".zip")
            Else
                System.IO.Compression.ZipFile.CreateFromDirectory(ASSYname, 
                  ASSYname + ".zip")
            End If
            MsgBox("Complete",, "Packed and Zipped")


            'ZipFile.CreateFromDirectory(ASSYname, ASSYname + " IR " + ".zip")

            Return True
            End If

            Return False
        End Function

    Shared Function PackandGo2()

        Dim swApp As SldWorks.SldWorks
        Dim swDoc As ModelDoc2
        Dim swPackAndGo As PackAndGo
        Dim swPackAndGo_Zip As PackAndGo

        Dim status As Boolean
        Dim pgFileNames As Object = Nothing
        Dim pgFileStatus As Object = Nothing
        Dim statuses As Object
        Dim Open_Docs As Object
        Dim Filename = String.Empty
        Dim Pathname = String.Empty
        Dim ASSYname = String.Empty
        Dim Pack_N_Go_name = String.Empty
        Dim MyFolder = String.Empty
        Dim ExistingFile = String.Empty

        Dim swDocType As Integer
        Dim namesCount As Integer


        Dim TimeNow As DateTime = DateTime.Now
        Dim format As String = "MM-dd-yyyy HH-mm-ss"

        swApp = CreateObject("SldWorks.Application")
        swApp.Visible = True
        swDoc = swApp.ActiveDoc

        'If swDoc Is Nothing Then
        If swDoc.GetType <> 3 Then

            Functions.Error_Form()
        Else

            swDoc.ForceRebuild3(True)
            swDoc.Save3(1, 1, 2)

            'check for existing file is true
            ExistingFile = swDoc.GetPathName
            If ExistingFile = "" Then
                Functions.Error_Form("File Error", "File is not Saved",, "Please 
                  save before using Pack n Go",,,)
            Else


                Filename = System.IO.Path.GetFileNameWithoutExtension
                  (ExistingFile)
                Pathname = System.IO.Path.GetDirectoryName(ExistingFile)
                ASSYname = Pathname + "\" + Filename + " (Pack and Go)"
                Pack_N_Go_name = Filename + " (Pack and Go)"

                swPackAndGo = swDoc.Extension.GetPackAndGo
                swPackAndGo.IncludeDrawings = True
                swPackAndGo.IncludeSimulationResults = False
                swPackAndGo.IncludeSuppressed = True
                swPackAndGo.IncludeToolboxComponents = True
                swPackAndGo.FlattenToSingleFolder = True

                swPackAndGo_Zip = swDoc.Extension.GetPackAndGo
                swPackAndGo_Zip.IncludeDrawings = True
                swPackAndGo_Zip.IncludeSimulationResults = False
                swPackAndGo_Zip.IncludeSuppressed = True
                swPackAndGo_Zip.IncludeToolboxComponents = True
                swPackAndGo_Zip.FlattenToSingleFolder = True

                namesCount = swPackAndGo.GetDocumentNamesCount


```

```vb
                status = swPackAndGo.GetDocumentSaveToNames(pgFileNames, pgFileStatus)




                Dim External_Files As New List(Of String)
                Dim exclude = {".slddrw", ".sldasm", ".sldprt"}
                Dim Extension As String

                For Each myFile As String In Directory.GetFiles(Pathname)
                    Extension = LCase(Path.GetExtension(myFile))
                    If Extension = exclude(0) Or Extension = exclude(1) Or Extension = exclude(2) Then
                        'MsgBox("failed")
                    Else
                        External_Files.Add(myFile)
                        'MsgBox(myFile)
                    End If

                Next
                status = swPackAndGo.AddExternalDocuments(External_Files.ToArray)

                'status = swPackAndGo_zip.AddExternalDocuments(External_Files.ToArray)


                'Puts Pack and Go files in native Directory
                status = swPackAndGo.SetSaveToName(True, Pathname)
                statuses = swDoc.Extension.SavePackAndGo(swPackAndGo)


                'Puts Pack and Go files in native Directory

                'swPackAndGo_Zip.AddPrefix = Filename + " - "
                'status = swPackAndGo_Zip.SetSaveToName(True, ASSYname)
                'statuses = swDoc.Extension.SavePackAndGo(swPackAndGo_Zip)



                Open_Docs = Opened_Docs()

                If swDoc.GetType = CInt(3) Then

                    swDocType = swDoc.GetType

                    For i = LBound(Open_Docs) To UBound(Open_Docs)
                        swDoc = Open_Docs(i)
                        Dim F_Name As String
                        Dim errors As Integer
                        Dim MoreErr As Integer

                        If swDoc.GetType = 2 Or swDoc.GetType = 3 Then
                            F_Name = SWFunctions.SW_FileName(swDoc.GetPathName())
                            swApp.IActivateDoc3(F_Name, False, errors)
                            swDoc.Save3(1, errors, MoreErr)
                            swApp.QuitDoc(F_Name)

                        End If

                    Next



                    Dim pgFileNames2 As Object
                    Dim pgFileStatus2 As Object

                    'disconnect error when creating folder of assyname june 15th, 2021.
                    'Resolved, can't close document before finishing pack and go

                    status = swPackAndGo_Zip.GetDocumentSaveToNames(pgFileNames2, pgFileStatus2)

                    status = swPackAndGo_Zip.AddExternalDocuments(External_Files.ToArray)

                    status = swPackAndGo_Zip.SetSaveToName(True, ASSYname + ".zip")
                    swPackAndGo_Zip.AddPrefix = Filename + " - "



                    'ReDim pgFileNames2(namesCount - 1)
                    'ReDim pgFileStatus2(namesCount - 1)
                    'status = swPackAndGo_Zip.GetDocumentSaveToNames(pgFileNames2, pgFileStatus2)
                    'Debug.Print("")
                    'Debug.Print("  My Pack and Go path and filenames after adding prefix and suffix: ")
                    'For i = 0 To (namesCount - 1)
                    '   Debug.Print("    My path and filename is: " & pgFileNames2(i))
                    'Next i

                    Dim Zipped_File As String
                    Zipped_File = ASSYname + ".zip"

                    If File.Exists(Zipped_File) Then

                        status = swPackAndGo_Zip.SetSaveToName(True, ASSYname + " - " + TimeNow.ToString(format) + ".zip")
```

```
                            statuses = swDoc.Extension.SavePackAndGo(swPackAndGo_Zip)
                        Else

                            statuses = swDoc.Extension.SavePackAndGo(swPackAndGo_Zip)
                        End If
                        swApp.QuitDoc(ExistingFile)
                        swApp.OpenDoc6(ExistingFile, swDocType, 1, "", 1, 2)

                    End If

                    'disconnect error when creating folder of assyname june 15th, 2021.
                    'If File.Exists(ASSYname + ".zip") Then

                    '   System.IO.Compression.ZipFile.CreateFromDirectory(ASSYname, ASSYname + " - " + TimeNow.ToString(format) + ".zip")
                    'Else
                    '   System.IO.Compression.ZipFile.CreateFromDirectory(ASSYname, ASSYname + ".zip")
                    'End If
                    'MsgBox("Complete",, "Packed and Zipped")

                    'IDK where this came from
                    'ZipFile.CreateFromDirectory(ASSYname, ASSYname + " IR " + ".zip")

                    Return True
                End If
            End If

            Return False
        End Function

        Shared Function Save_Step()

                Dim swApp As SldWorks.SldWorks
                Dim swDoc As ModelDoc2
                Dim status As Boolean = False

                swApp = CreateObject("SldWorks.Application")
                swDoc = swApp.ActiveDoc

                If swDoc Is Nothing Then

                    Functions.Error_Form()

                Else

                    If swDoc.GetType = 1 Then
                    'get directory path, sometimes it doesn't save in the correct
```

```
                      folder
                Dim PathName = String.Empty
                    PathName = swDoc.GetPathName()
                    PathName = SW_FileName(PathName)
                    swDoc.ClearSelection2(True)
                    status = swDoc.SaveAs(PathName + ".STEP")

                Else
                    Functions.Error_Form("Not a Part File", "This is not a Part 
                     File")
                End If
            End If

            Return status
        End Function


    Shared Function Save_As()

            Dim swApp As SldWorks.SldWorks
            Dim swDoc As ModelDoc2
            Dim status As Boolean = False
            Dim bool As Boolean

            swApp = CreateObject("SldWorks.Application")
            swDoc = swApp.ActiveDoc

            If swDoc Is Nothing Then

                Functions.Error_Form()

            Else

                Dim OG_Path As String = swDoc.GetPathName()
                bool = swDoc.Extension.RunCommand
                  (swCommands_e.swCommands_SaveAs, "")
                Dim New_Path As String = swDoc.GetPathName()

                If OG_Path = New_Path Then
                    status = False
                Else
                    status = True
                End If

            End If

            Return status
        End Function


    Shared Function Save_Doc()
            Dim swApp As SldWorks.SldWorks
```

```vb
            Dim bool As Boolean
            Dim swDoc As ModelDoc2

            swApp = CreateObject("SldWorks.Application")
            swDoc = swApp.ActiveDoc

            bool = swDoc.Extension.RunCommand(swCommands_e.swCommands_SaveAs, "")
            Return bool
        End Function


    Shared Function View_Scale(Outline1() As Double, Optional Outline2() As Double = Nothing, Optional Outline3() As Double = Nothing)

            Dim View1 As Double = 10
            Dim View2 As Double = 10
            Dim View3 As Double = 10
            Dim Small_Scale As Double = 10 'Set as high value

            View1 = Boundary_box(Outline1)

            If Outline2 IsNot Nothing Then
                View2 = Boundary_box(Outline2)
            End If

            If Outline3 IsNot Nothing Then
                View3 = Boundary_box(Outline3)
            End If

            If View1 <> 1 Or View2 <> 1 Or View3 <> 1 Then
                Dim Scales As Double() = {View1, View2, View3}

                For Each element As Double In Scales
                    Small_Scale = Math.Min(Small_Scale, element)
                Next
                'MsgBox(View1 & ", " & View2 & ", " & View3 & ", " & Small_Scale)

            End If

            Small_Scale = Math.Round(1 / Small_Scale, 1)
            'MsgBox(Small_Scale & " This is the scale value")

            'If Small_Scale <= 0.5 Then

            '   Small_Scale = Math.Round(1 / Small_Scale, 0)
            '   MsgBox(Small_Scale)

            'ElseIf Small_Scale > 0.25 Then
            '   MsgBox(Small_Scale)
            '   Small_Scale = (1 / Small_Scale) * 2
            '   MsgBox(Small_Scale)
            '   Small_Scale = Math.Round(Small_Scale, 1, MidpointRounding.AwayFromZero)
            '   MsgBox(Small_Scale)
            '   Small_Scale = Math.Floor(Small_Scale)
            '   'MsgBox(Small_Scale)
            '   'Small_Scale = Small_Scale / 2
            '   MsgBox(Small_Scale)

            'End If

            Return Small_Scale
        End Function


    Shared Function Boundary_box(Outline() As Double)

            Dim View_Scale_Factor As Decimal() = {0.0, 0.0}
            Dim Boundary As Double() = {0.0, 0.0, 0.0, 0.0}
            Dim Scales_sw As Double() = {1 / 2, 1 / 3, 1 / 4, 1 / 5, 1 / 6, 1 / 7, 1 / 8, 1 / 10, 1 / 12, 1, 2 / 3, 2, 3}
            Dim Scale_Factor As Double = 0.0

            Boundary = Outline

            Boundary(0) = Boundary(0) * 39.3701 ' x min
            Boundary(1) = Boundary(1) * 39.3701 ' y min
            Boundary(2) = Boundary(2) * 39.3701 ' x max
            Boundary(3) = Boundary(3) * 39.3701 ' y max

            Boundary(2) = Boundary(2) - Boundary(0)
            Boundary(3) = Boundary(3) - Boundary(1)

            View_Scale_Factor(0) = 4.875 / Boundary(3) 'Scale factor to achieve 4.875" on Y height
            View_Scale_Factor(1) = 10.5 / Boundary(2) 'Scale factor to achieve 10.5" on X width

            'MsgBox(View_Scale_Factor(0) & " Y - " & View_Scale_Factor(1))

            Dim smallestdiff As Double = Math.Abs(View_Scale_Factor(0) - Scales_sw(0))
            Dim smallestdiffIndex = 0
            Dim Currdiff As Double
            Dim j = 0
            For j = 0 To Scales_sw.Count - 1
                Currdiff = Math.Abs(View_Scale_Factor(0) - Scales_sw(j))
                If Currdiff < smallestdiff Then
                    smallestdiff = Currdiff
                    smallestdiffIndex = j
                End If

            Next

            View_Scale_Factor(0) = Scales_sw(smallestdiffIndex)

            For j = 0 To Scales_sw.Count - 1
                Currdiff = Math.Abs(View_Scale_Factor(1) - Scales_sw(j))
                If Currdiff < smallestdiff Then
                    smallestdiff = Currdiff
                    smallestdiffIndex = j
                End If

            Next

            View_Scale_Factor(1) = Scales_sw(smallestdiffIndex)

            'Compare each View_Scale_Factor to common scale options, pick the 
              one closest

            'If Boundary(3) > 10 Then
            '    'scale to make Boundary_Box_Size(3)=8.25
            '    View_Scale_Factor(0) = (10 / Boundary(3))
            '    MsgBox(View_Scale_Factor(0))
            '    View_Scale_Factor(0) = Math.Round(View_Scale_Factor(0), 1)
            '    MsgBox(View_Scale_Factor)
            'Else
            '    View_Scale_Factor(0) = 1
            'End If

            Scale_Factor = Math.Min(View_Scale_Factor(0), View_Scale_Factor(1))

            Return Scale_Factor
        End Function


    Shared Function Get_View_Info(sw_View As View) 'Send a View in the first 
      view format

            Dim Array_Views As New List(Of Object)()
            Dim i = 0
            Dim Views As View
            Dim Test As View

            Views = sw_View
            'sw_View is the sheet view
            While Views IsNot Nothing

                Views = Views.GetNextView

                MsgBox(Views.Name)
                Array_Views.Add(Views)

                Test = Array_Views(i)
```

```
                    MsgBox(Test.Name)
                    'sw_View = sw_View.GetNextView
                    i += 1

                End While


                Return Array_Views

            End Function


        'Add_NoteInfo is obsolete use Add_NoteInfo2
        Shared Function Add_NoteInfo(swDoc As ModelDoc2, Assembly_Docs As Integer, 
          Part_Docs As Integer, Drawing_View As String,
                                         File_name As String, Instance_Num As List(Of 
                          String))

                Dim sw_View As View
                Dim swNote As Note
                Dim swAnno As Annotation

                Dim Text_Add As String
                Dim Const_Text As String

                Dim Files As Integer = Assembly_Docs + Part_Docs
                Dim x_pos As Double = 0.0
                Dim y_pos As Double = 0.0
                Dim z_pos As Double = 0.0

                Dim ChartoTrim As Char() = {"@"}

                sw_View = swDoc.GetFirstView
                sw_View = sw_View.GetNextView

                Const_Text = "$PRPSMODEL:" & Chr(34) & "NOTE INFO" & Chr(34) & " 
                  $COMP:" & Chr(34) & File_name & "@" & Drawing_View & "/"

                For i = 0 To Instance_Num.Count  'Files - 1
                    If i = 0 Then

                        Text_Add = "$PRPSMODEL:" & Chr(34) & "NOTE INFO" & Chr(34) & 
                         " $COMP:" & Chr(34) & File_name & "-1" & "@" & Drawing_View 
                         & Chr(34)
                        swNote = swDoc.InsertNote(Text_Add)

                    Else

                        Text_Add = Const_Text + Instance_Num(i - 1) & Chr(34)
                        swNote = swDoc.InsertNote(Text_Add)

                    End If
```

```vb

                swAnno = swNote.GetAnnotation()
                swAnno.SetAttachedEntities(sw_View)
                swAnno.SetPosition2(x_pos, y_pos, z_pos)
                y_pos -= 0.00635

            Next

            swDoc.WindowRedraw()

            Return True

        End Function


    Shared Function Add_NoteInfo2(swDoc As ModelDoc2, Drawing_View As String,
                                 File_name As String, Instance_Num As List(Of
            New_Drawing.SW_Opened_Files), Optional ByVal Add_Bom As
            Boolean = False)

        Dim sw_View As View
        Dim swNote_Info As Note
        Dim swAnno_Info As Annotation
        Dim swNote_DESCRIPTION As Note = Nothing
        Dim swAnno_DESCRIPTION As Annotation
        Dim swNote_NOMENCLATURE As Note = Nothing
        Dim swAnno_NOMENCLATURE As Annotation
        Dim swNote_SPEC As Note = Nothing
        Dim swAnno_SPEC As Annotation

        Dim swLayer As Layer
        Dim swLayerMgr As LayerMgr


        Dim Text_Add As String
        Dim NOTE_INFO As String
        Dim DESCRIPTION As String
        Dim NOMENCLATURE As String
        Dim SPEC As String
        Dim Bool As Integer

        Dim x_pos As Double = 0.0
        Dim x_pos_Nom As Double = 0.1
        Dim x_pos_Des As Double = 0.2
        Dim x_pos_Spec As Double = 0.3
        Dim y_pos As Double = 0
        Dim z_pos As Double = 0.0

        Dim ChartoTrim As Char() = {"@"}

        'swLayer =
        swLayerMgr = swDoc.GetLayerManager
        Bool = swLayerMgr.SetCurrentLayer("NOTES")
        'swLayer = swLayerMgr.GetLayer("NOTES")


        sw_View = swDoc.GetFirstView
            sw_View = sw_View.GetNextView

        NOTE_INFO = "$PRPSMODEL:" & Chr(34) & "NOTE INFO" & Chr(34) & " $COMP:"
          & Chr(34) & File_name & "@" & Drawing_View & "/"

        DESCRIPTION = "$PRPSMODEL:" & Chr(34) & "DESCRIPTION" & Chr(34) & "
          $COMP:" & Chr(34) & File_name & "@" & Drawing_View & "/"
            NOMENCLATURE = "$PRPSMODEL:" & Chr(34) & "NOMENCLATURE" & Chr(34) &
              " $COMP:" & Chr(34) & File_name & "@" & Drawing_View & "/"
            SPEC = "$PRPSMODEL:" & Chr(34) & "SPEC" & Chr(34) & " $COMP:" & Chr
              (34) & File_name & "@" & Drawing_View & "/"

            For i = 0 To Instance_Num.Count  'Files - 1

                If i = 0 Then

                    Text_Add = "$PRPSMODEL:" & Chr(34) & "NOTE INFO" & Chr(34) &
                     " $COMP:" & Chr(34) & File_name & "-1" & "@" & Drawing_View
                     & Chr(34)
                    swNote_Info = swDoc.InsertNote(Text_Add)

                    If New_Drawing.Add_BOM_Hardware.Checked = True Then
                        Text_Add = "$PRPSMODEL:" & Chr(34) & "DESCRIPTION" & Chr
                    (34) & " $COMP:" & Chr(34) & File_name & "-1" & "@" &
                    Drawing_View & Chr(34)
                        swNote_DESCRIPTION = swDoc.InsertNote(Text_Add)

                        Text_Add = "$PRPSMODEL:" & Chr(34) & "NOMENCLATURE" &
                    Chr(34) & " $COMP:" & Chr(34) & File_name & "-1" & "@" &
                    Drawing_View & Chr(34)
                        swNote_NOMENCLATURE = swDoc.InsertNote(Text_Add)

                        Text_Add = "$PRPSMODEL:" & Chr(34) & "SPEC" & Chr(34) &
                    " $COMP:" & Chr(34) & File_name & "-1" & "@" & Drawing_View
                    & Chr(34)
                        swNote_SPEC = swDoc.InsertNote(Text_Add)
                    End If
                Else

                    Text_Add = NOTE_INFO + Instance_Num(i - 1).Instance_ID & Chr
                    (34)
                    'MsgBox(Instance_Num(i - 1).Instance_ID)
                    swNote_Info = swDoc.InsertNote(Text_Add)

                    If Add_Bom = True Then

                        Text_Add = DESCRIPTION + Instance_Num(i - 1).Instance_ID
```

```
                      & Chr(34)
                        swNote_DESCRIPTION = swDoc.InsertNote(Text_Add)

                        Text_Add = NOMENCLATURE + Instance_Num(i - 1).Instance_ID & Chr(34)
                        swNote_NOMENCLATURE = swDoc.InsertNote(Text_Add)

                        Text_Add = SPEC + Instance_Num(i - 1).Instance_ID & Chr(34)
                        swNote_SPEC = swDoc.InsertNote(Text_Add)
                    End If
                End If

                swAnno_Info = swNote_Info.GetAnnotation()
                swAnno_Info.SetAttachedEntities(sw_View)
                swAnno_Info.SetPosition2(x_pos, y_pos, z_pos)

                'If New_Drawing.Add_BOM_Hardware.Checked = True Then
                If Add_Bom = True And i <> 0 Then
                    swAnno_NOMENCLATURE = swNote_NOMENCLATURE.GetAnnotation()
                    swAnno_NOMENCLATURE.SetAttachedEntities(sw_View)
                    swAnno_NOMENCLATURE.SetPosition2(x_pos_Nom, y_pos, z_pos)

                    swAnno_DESCRIPTION = swNote_DESCRIPTION.GetAnnotation()
                    swAnno_DESCRIPTION.SetAttachedEntities(sw_View)
                    swAnno_DESCRIPTION.SetPosition2(x_pos_Des, y_pos, z_pos)

                    swAnno_SPEC = swNote_SPEC.GetAnnotation()
                    swAnno_SPEC.SetAttachedEntities(sw_View)
                    swAnno_SPEC.SetPosition2(x_pos_Spec, y_pos, z_pos)
                End If


                y_pos -= 0.00889

            Next

            'swDoc.WindowRedraw()

            Return True

        End Function


    Shared Function Add_Docs(ByVal swComp As Component2, ByVal nLevel As Integer)

        'try swApp.GetDocumentDependencies2 method

        swApp = CreateObject("SldWorks.Application")

        Dim swPart As ModelDoc2
```

```vb
        Dim swPart2 As ModelDoc2
        Dim swPart3 As ModelDoc2
        Dim swPart4 As ModelDoc2
        Dim swChildComp As Component2
        Dim swParent As Component2

        Dim vChildComp As Object

        Dim Status As Boolean = False
        Dim Used As Boolean = False
        Dim Add_To_Part As Boolean = False
        Dim Add_To_List As Boolean = False
        Dim isAssy As Boolean = False
        Dim isPart As Boolean = False


        Dim ValOut = String.Empty
        Dim Dash_XX = String.Empty
        Dim wasResolved As Boolean
        Dim linkToProp As Boolean
        Dim Dash_Name = String.Empty
        Dim Temp_Name = String.Empty

        Dim errorval As Integer

        Dim CusProp As String() = {"PART NUMBER", "NOMENCLATURE", "DESCRIPTION", "SPEC", "MATERIAL", "WEIGHT"}
        Dim RecAssyCusProp As String() = {"N/A", "N/A", "N/A", "N/A", "N/A", "N/A"}


        If nLevel = 1 Then

            'swAssy_Docs.Clear()
            'swPart_Docs.Clear()
            'swDwg_Docs.Clear()
            swComp_Assy = swComp.Name2
            swPart2 = swComp.GetModelDoc2

            If swPart2.GetType = swDocumentTypes_e.swDocASSEMBLY Then

                swModelDocExt_Assy = swPart2.Extension
                CusProperties_Assy = swModelDocExt_Assy.CustomPropertyManager("")

                For propNum = 0 To UBound(CusProp)
                    Dash_Name = CusProperties_Assy.Get6(CusProp(propNum), True, ValOut, RecAssyCusProp(propNum), wasResolved, linkToProp)
                Next

                If CusProperties_Assy.Get6(CusProp(0), True, ValOut, RecAssyCusProp(0), wasResolved, linkToProp) = 2 Then
```

```
                    If RecAssyCusProp(0) <> "" And RecAssyCusProp(0) <> "-XX" Then

                        Temp_Name = RecAssyCusProp(0).Substring(0, 1)

                        If Temp_Name = "-" Then
                            Add_To_List = True
                        End If
                    End If

                End If
                If Add_To_List = True Then
                    swAssy_Docs.Add(New Assy_Docs(swComp_Assy, "Null", swComp.GetSelectByIDString(), RecAssyCusProp(0), RecAssyCusProp(1), RecAssyCusProp(2), RecAssyCusProp(3), RecAssyCusProp(4), RecAssyCusProp(5)))
                End If

            End If
        End If

        vChildComp = swComp.GetChildren
        For i = 0 To UBound(vChildComp)

            Used = False
            Dim pString = String.Empty
            Dim aString = String.Empty
            Dim Part_To_Open = String.Empty
            Dim RecCusProp = {"N/A", "N/A", "N/A", "N/A", "N/A", "N/A"}

            swChildComp = vChildComp(i)

            If swChildComp.IsSuppressed() = False Then
                Add_To_List = False
                swParent = swChildComp.GetParent

                If swParent Is Nothing Then
                    swPart = swChildComp.GetModelDoc2
                    aString = swChildComp.Name2
                    'MsgBox("Assy - " + aString)

                    If swPart.GetType = 2 Then
                        isAssy = True
                        aString = aString.Substring(0, aString.LastIndexOf("-"))
                    ElseIf swPart.GetType = 1 Then
                        isPart = True
                        pString = aString.Substring(0, aString.LastIndexOf("-"))

                    End If

                Else

```

```
                    Dim tString As String
                    swPart4 = swChildComp.GetModelDoc2
                    swPart = swParent.GetModelDoc2
                    If swPart4.GetType = 2 Then
                        'MsgBox("Assy ")
                        isAssy = True
                    ElseIf swPart4.GetType = 1 Then
                        'MsgBox("Part")
                        isPart = True
                    End If

                    tString = swParent.Name2.Substring(0, swParent.Name2.LastIndexOf("-"))
                    'aString = swChildComp.Name2

                    'If swPart.GetType = 2 Then
                    '   isAssy = True
                    '   aString = aString.Substring(0, aString.LastIndexOf("-"))
                    'ElseIf swPart.GetType = 1 Then
                    '   isPart = True
                    '   pString = aString.Substring(0, aString.LastIndexOf("-"))

                    'End If
                    pString = swChildComp.Name2
                    'MsgBox("Part - " + pString)

                    While pString.IndexOf("/") <> -1
                        pString = pString.Substring(pString.IndexOf("/") + 1)
                    End While

                    pString = pString.Substring(0, pString.LastIndexOf("-"))

                End If



                If pString IsNot "" Then

                    If swPart_Docs.Count > 0 Then
                        For f = 0 To swPart_Docs.Count - 1
                            If swPart_Docs(f).subcomp = pString Then
                                Used = True
                                isAssy = False
                                isPart = False
                            End If
                        Next
                    End If

                    If Used = False Then

                        swPart3 = swApp.ActivateDoc3(pString, 0, 1, errorval)

```

```vb
                If swPart3.GetType = swDocumentTypes_e.swDocPART Then

                    swModelDocExt_Part = swPart3.Extension
                    CusProperties_Part = swModelDocExt_Part.CustomPropertyManager("")

                    For propNum = 0 To UBound(CusProp)
                        Dash_Name = CusProperties_Part.Get6(CusProp(propNum), True, ValOut, RecCusProp(propNum), wasResolved, linkToProp)
                    Next

                    If CusProperties_Assy.Get6(CusProp(0), True, ValOut, RecAssyCusProp(0), wasResolved, linkToProp) = 2 Then
                        If RecCusProp(0) <> "" And RecCusProp(0) <> "-XX" Then

                            Temp_Name = RecCusProp(0).Substring(0, 1)

                            If Temp_Name = "-" Then
                                Add_To_List = True
                            End If
                        End If

                    End If

                ElseIf swPart3.GetType = swDocumentTypes_e.swDocASSEMBLY Then

                    swModelDocExt_Assy = swPart3.Extension
                    CusProperties_Assy = swModelDocExt_Assy.CustomPropertyManager("")

                    For propNum = 0 To UBound(CusProp)
                        Dash_Name = CusProperties_Assy.Get6(CusProp(propNum), True, ValOut, RecCusProp(propNum), wasResolved, linkToProp)
                    Next

                    If CusProperties_Assy.Get6(CusProp(0), True, ValOut, RecAssyCusProp(0), wasResolved, linkToProp) = 2 Then
                        If RecCusProp(0) <> "" And RecCusProp(0) <> "-XX" Then

                            Temp_Name = RecCusProp(0).Substring(0, 1)

                            If Temp_Name = "-" Then
                                Add_To_List = True
                            End If
                        End If

                    End If
```

```vb

                            End If
                            swApp.CloseDoc(pString)
                        End If


                    ElseIf swPart.GetType = swDocumentTypes_e.swDocPART Then

                        swModelDocExt_Part = swPart.Extension
                        CusProperties_Part = swModelDocExt_Part.CustomPropertyManager("")

                        For propNum = 0 To UBound(CusProp)
                            Dash_Name = CusProperties_Part.Get6(CusProp(propNum), True, ValOut, RecCusProp(propNum), wasResolved, linkToProp)
                        Next

                        If CusProperties_Assy.Get6(CusProp(0), True, ValOut, RecAssyCusProp(0), wasResolved, linkToProp) = 2 Then
                            If RecCusProp(0) <> "" And RecCusProp(0) <> "-XX" Then

                                Temp_Name = RecCusProp(0).Substring(0, 1)

                                If Temp_Name = "-" Then
                                    Add_To_List = True
                                End If
                            End If

                        End If

                    ElseIf swPart.GetType = swDocumentTypes_e.swDocASSEMBLY Then

                        swModelDocExt_Assy = swPart.Extension
                        CusProperties_Assy = swModelDocExt_Assy.CustomPropertyManager("")

                        For propNum = 0 To UBound(CusProp)
                            Dash_Name = CusProperties_Assy.Get6(CusProp(propNum), True, ValOut, RecCusProp(propNum), wasResolved, linkToProp)
                        Next

                        If CusProperties_Assy.Get6(CusProp(0), True, ValOut, RecAssyCusProp(0), wasResolved, linkToProp) = 2 Then
                            If RecCusProp(0) <> "" And RecCusProp(0) <> "-XX" Then

                                Temp_Name = RecCusProp(0).Substring(0, 1)

                                If Temp_Name = "-" Then
                                    Add_To_List = True
                                End If
                            End If

                        End If

                    End If

                    Add_To_Part = True

                    If isAssy = True And Add_To_List = True Then
                        isAssy = False
                        If swPart.GetType = swDocumentTypes_e.swDocASSEMBLY Then
                            swAssy_Docs.Add(New Assy_Docs(swComp_Assy, aString, swChildComp.GetSelectByIDString(), RecCusProp(0), RecCusProp(1),
                                                          RecCusProp(2), RecCusProp(3), RecCusProp(4), RecCusProp(5)))
                        End If
                    End If

                    If isPart = True And Add_To_List = True Then
                        isPart = False
                        If swPart_Docs.Count = 0 Then
                            swPart_Docs.Add(New Part_Docs(swComp_Assy, pString, swChildComp.GetSelectByIDString(), RecCusProp(0), RecCusProp(1),
                                                          RecCusProp(2), RecCusProp(3), RecCusProp(4), RecCusProp(5)))
                            Add_To_Part = False
                            'Else
                            '   For q = 0 To swPart_Docs.Count - 1
                            '       If swPart_Docs(q).subcomp = pString Then
                            '           Add_To_Part = False
                            '           Exit For
                            '       End If
                            '   Next
                        End If
                        If Add_To_Part = True Then
                            swPart_Docs.Add(New Part_Docs(swComp_Assy, pString, swChildComp.GetSelectByIDString(), RecCusProp(0), RecCusProp(1),
                                                          RecCusProp(2), RecCusProp(3), RecCusProp(4), RecCusProp(5)))

                            Add_To_Part = False
                        End If
                    End If
                    'Debug.Print(swChildComp.Name2)
                    Status = Add_Docs(swChildComp, nLevel + 1)
                End If
            Next i

            Return True

        End Function


        Shared Function Compare()
            Dim status As Boolean = True

            Return status
        End Function

        Shared Function Out_Put()

            If swAssy_Docs.Count > 0 Then

                swAssy_Docs = swAssy_Docs.OrderByDescending(Function(x) x.Part_Number).ToList

                'For z = 0 To swAssy_Docs.Count - 1
                '    'Debug.Print("Assembly - " & swAssy_Docs(z).instance_ID)

                '    If swAssy_Docs(z).Part_Number <> "" Then

                '        If swAssy_Docs(z).Part_Number.Substring(0, 1) = "-" And swAssy_Docs(z).Part_Number <> "-XX" Then

                '            Debug.Print("Assembly - " & swAssy_Docs(z).Comp & " :- " & swAssy_Docs(z).subcomp & " :- " &
                '                        " --- " & swAssy_Docs(z).Part_Number & " , " & swAssy_Docs(z).Nomenclature &
                  '                        " , " & swAssy_Docs(z).Description & " , " & swAssy_Docs(z).Spec & " , " & swAssy_Docs(z).Material &
                '                        " , " & swAssy_Docs(z).Weight)
                '            '&
                '            'System.Environment.NewLine & " " & swAssy_Docs(z).instance_ID)

                '        End If
                '    End If
                'Next
            End If

            Debug.Print(System.Environment.NewLine & System.Environment.NewLine)

            If swPart_Docs.Count > 0 Then

                swPart_Docs = swPart_Docs.OrderByDescending(Function(x) x.Part_Number).ToList

                'For z = 0 To swPart_Docs.Count - 1
                '    'MsgBox(swPart_Docs(z).Part_Number + "  -  " + swPart_Docs(z).Comp + "  -  " + swPart_Docs(z).subcomp)
                '    If swPart_Docs(z).Part_Number <> "" Then

                '        If swPart_Docs(z).Part_Number.Substring(0, 1) = "-" And
```

```vb
                        swPart_Docs(z).Part_Number <> "-XX" Then

            '           Debug.Print("Part - " & swPart_Docs(z).Comp & " :- " & swPart_Docs(z).subcomp & " :-: " &
            '                   " --- " & swPart_Docs(z).Part_Number & " , " & swPart_Docs(z).Nomenclature &
            '                   " , " & swPart_Docs(z).Description & " , " & swPart_Docs(z).Spec & " , " & swPart_Docs(z).Material &
            '                   " , " & swPart_Docs(z).Weight) ' &
            '           'System.Environment.NewLine & " " & swPart_Docs(z).instance_ID)

            '       End If
            '   End If
            'Next
        End If

        If swDwg_Docs.Count > 0 Then

            For z = 0 To swDwg_Docs.Count - 1
                Debug.Print("Drawing - " & swDwg_Docs(z).Comp)
            Next
        End If


        Return True

    End Function

    Shared Function Docs_To_Excel(Path As String, Filename As String)

        Dim xlApp As Microsoft.Office.Interop.Excel.Application
        Dim xlWorkBook As Microsoft.Office.Interop.Excel.Workbook
        Dim xlWorkSheet As Microsoft.Office.Interop.Excel.Worksheet
        Dim Range As Microsoft.Office.Interop.Excel.Range
        Dim misValue As Object = System.Reflection.Missing.Value

        Dim i As Integer
        Dim j As Integer
        Dim Assy_row As Integer = 3
        Dim Assy_col As Integer = 1
        Dim Part_row As Integer = Assy_row + swAssy_Docs.Count + 2
        Dim Part_col As Integer = 1

        xlApp = New ApplicationClass

        Dim Assy_Array(swAssy_Docs.Count - 1, 8) As String
        Dim Assy_Count = 0

        Dim Part_Array(swPart_Docs.Count - 1, 8) As String
        Dim Part_Count = 0

        For Each Stringitem As Assy_Docs In swAssy_Docs

            Assy_Array(Assy_Count, 0) = "Assembly"
            Assy_Array(Assy_Count, 1) = Stringitem.Comp
            Assy_Array(Assy_Count, 2) = Stringitem.subcomp
            Assy_Array(Assy_Count, 3) = Stringitem.Part_Number
            Assy_Array(Assy_Count, 4) = Stringitem.Nomenclature
            Assy_Array(Assy_Count, 5) = Stringitem.Description
            Assy_Array(Assy_Count, 6) = Stringitem.Spec
            Assy_Array(Assy_Count, 7) = Stringitem.Material
            Assy_Array(Assy_Count, 8) = Stringitem.Weight

            Assy_Count = +1

        Next

        For Each Stringitem As Part_Docs In swPart_Docs

            Part_Array(Part_Count, 0) = "Part"
            Part_Array(Part_Count, 1) = Stringitem.Comp
            Part_Array(Part_Count, 2) = Stringitem.subcomp
            Part_Array(Part_Count, 3) = Stringitem.Part_Number
            Part_Array(Part_Count, 4) = Stringitem.Nomenclature
            Part_Array(Part_Count, 5) = Stringitem.Description
            Part_Array(Part_Count, 6) = Stringitem.Spec
            Part_Array(Part_Count, 7) = Stringitem.Material
            Part_Array(Part_Count, 8) = Stringitem.Weight

            Part_Count += 1

        Next

        xlWorkBook = xlApp.Workbooks.Open("T:\Engineering\Non-Site Specific
          \PARTS\SW Macros\Model Creator\Resources\Part Numbers.xlsx")
        xlApp.WindowState = XlWindowState.xlMaximized
        xlApp.Visible = True


        xlWorkSheet = xlWorkBook.Sheets(1)

        With xlWorkSheet
            Range = .Range(.Cells(Assy_row, Assy_col), .Cells(UBound(Assy_Array,
              1) - LBound(Assy_Array, 1) + Assy_row, UBound(Assy_Array, 2) -
              LBound(Assy_Array, 2) + Assy_col))
        End With
        Range.Value = Assy_Array

        With xlWorkSheet
            Range = .Range(.Cells(Part_row, Part_col), .Cells(UBound(Part_Array,
              1) - LBound(Part_Array, 1) + Part_row, UBound(Part_Array, 2) -
              LBound(Part_Array, 2) + Part_col))
        End With
        Range.Value = Part_Array

        Path = Path + "\" + Filename + " - Doc Data.xlsx"
        xlWorkBook.SaveAs(Path)
        'xlWorkBook.Close()

        Return True
    End Function

    Shared Function Name_Tables(swFeat_Table As Feature)

        Dim swGeneralTableFeature As GeneralTableFeature
        Dim swTableAnnotation As TableAnnotation
        Dim nbrTableAnnotations As Integer
        Dim Insert_WeightTable As Boolean = True
        Dim Insert_BOMTable As Boolean = True
        Dim Columns_max As Integer
        Dim tableAnnotations() As Object


        swGeneralTableFeature = swFeat_Table.GetSpecificFeature2
        nbrTableAnnotations = swGeneralTableFeature.GetTableAnnotationCount
        tableAnnotations = swGeneralTableFeature.GetTableAnnotations
        swTableAnnotation = tableAnnotations(0)
        Columns_max = swTableAnnotation.ColumnCount()
        If Columns_max = 4 Then

            swFeat_Table.Name = "WEIGHT TABLE"
            Insert_WeightTable = False
        ElseIf Columns_max > 4 Then

            swFeat_Table.Name = "PART LIST TABLE"
            Insert_BOMTable = False
        Else

        End If


        Return True
        'Return Insert_WeightTable, Insert_BOMTable



    End Function

End Class

```