```vb
Friend Module ModDisk
    Public DoOnErr As Boolean = True
    Public ErrCode As Integer = 0

    Public ReadOnly UserDeskTop As String =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop)
    Public ReadOnly UserFolder As String =
Environment.GetFolderPath(Environment.SpecialFolder.UserProfile)

    Public ReadOnly Ascii2Petscii As Byte() = My.Resources.Ascii2DisplayCode

    Public Enum PackerTypes As Byte
        Faster = 0
        Better = 1
    End Enum

    Public Packer As PackerTypes = PackerTypes.Faster

    'Public DiskLoop As Integer = 0

    Public Drive() As Byte
    Public DirBlocks(511) As Byte
    Public DirPtr(127) As Integer
    Public LastBitPtr As Integer
    Public LastBufferCnt As Integer

    Public TotLit, TotMatch As Integer

    'Public ReadOnly CustomIL As Boolean = True
    Public ReadOnly DefaultIL0 As Byte = 4
    Public ReadOnly DefaultIL1 As Byte = 3
    Public ReadOnly DefaultIL2 As Byte = 3
    Public ReadOnly DefaultIL3 As Byte = 3
    Public IL0 As Byte = DefaultIL0
    Public IL1 As Byte = DefaultIL1
    Public IL2 As Byte = DefaultIL2
    Public IL3 As Byte = DefaultIL3

    Public BufferCnt As Integer = 0

    Public FileUnderIO As Boolean = False
    Public IOBit As Byte

    Public ReadOnly SectorSkew As Integer = 2       'Skew if disk is formatted using Format II by TLR

    Public ReadOnly StdSectorsPerDisk As Integer = 664          'Standard disk
    Public ReadOnly StdTracksPerDisk As Integer = 35
    Public ReadOnly StdBytesPerDisk As Integer = 174848

    Public ReadOnly ExtSectorsPerDisk As Integer = StdSectorsPerDisk + 85     'Exnteded disk
    Public ReadOnly ExtTracksPerDisk As Integer = 40
    Public ReadOnly ExtBytesPerDisk As Integer = StdBytesPerDisk + (85 * 256)

    Public SectorsPerDisk As Integer = StdSectorsPerDisk
    Public TracksPerDisk As Integer = StdTracksPerDisk

    Public BlocksFree As Integer = SectorsPerDisk

    Public TabT(ExtSectorsPerDisk - 1), TabS(ExtSectorsPerDisk - 1), TabSCnt(ExtSectorsPerDisk - 1),
TabStartS(ExtTracksPerDisk) As Byte   'TabStartS is 1 based

    Public Disk(StdBytesPerDisk - 1), NextTrack, NextSector As Byte   'Next Empty Track and Sector
    Public MaxSector As Byte = 18, LastSector, Prg(), ReferenceFile() As Byte

    Public Track(40), CT, CS, CP, BlockCnt As Integer
    Public TestDisk As Boolean = False
    Public StartTrack As Byte = 1
    Public StartSector As Byte = 0

    Public ByteSt(), Buffer(255), LastByte, AdLo, AdHi As Byte
    Public Match, MaxBit, MatchSave(), PrgLen, Distant As Integer
    Public MatchOffset(), MatchCnt, RLECnt, MatchLen(), MaxSave, MaxOffset, MaxLen, LitCnt, BuffAdd,
PrgAdd As Integer
    Public MaxSLen, MaxSOff, MaxSSave As Integer
    Public DistAd(), DistLen(), DistSave(), DistCnt, DistBase As Integer
    Public DtPos, CmPos, CmLast, DtLen, MatchStart As Integer
    Public LastPO, LastMS As Integer    'save previous POffset and MatchStart positions to recompress
last block of bundle
    Public LastBitP, LastBytC, LastBitC As Integer
    Public MatchType(), MaxType As String

    Public LastByteCt As Integer = 255
    Public LastMOffset As Integer = 0
    Public LastMLen As Integer = 0
    Public LastPOffset As Integer = -1
    Public LastMType As String = ""

    Public PreMOffset As Integer = 0
    Public PostMOffset As Integer = 0

    Public BitsSaved As Integer = 0
    Public BytesSaved As Integer = 0

    Public PreOLMO As Integer = 0
    Public PostOLMO As Integer = 0

    Public Script As String
    Public ScriptHeader As String = "[Sparkle Loader Script]"
    Public ScriptName As String
    Public DiskNo As Integer

    Public D64Name As String = "" '= My.Computer.FileSystem.SpecialDirectories.MyDocuments

    Public ReadOnly DefaultDiskHeader As String = "demo disk " + Year(Now).ToString
    Public ReadOnly DefaultDiskID As String = "sprkl"
    Public ReadOnly DefaultDemoName As String = "demo"

    Public DiskHeader As String = DefaultDiskHeader
    Public DiskID As String = DefaultDiskID
    Public DemoName As String = DefaultDemoName
    Public DemoStart As String = ""
    Public LoaderZP As String = "02"

    'Hi-Score File variables
    Public HSFileName As String = ""
    Public HSFile() As Byte
    Public HSAddress As Integer = 0
    Public HSOffset As Integer = 0
    Public HSLength As Integer = 0
    Public bSaverPlugin As Boolean = False

    'Product ID - unique to build, same for all disks in build, $000000-$ffffff
    Public ProductID As Integer = 0

    'Disk system: 35 vs 40 tracks
    'Public TracksOnDisk As Integer = 35

    Public SystemFile As Boolean = False
    Public FileChanged As Boolean = False

    Public DiskCnt As Integer = -1
    Public BundleCnt As Integer = -1
    Public FileCnt As Integer = -1
    Public CurrentDisk As Integer = -1
    Public CurrentBundle As Integer = -1
    Public CurrentFile As Integer = -1
    Public CurrentScript As Integer = -1
    Public BundleNo As Integer = -1
    Public MaxBundleNoExceeded As Boolean = False

    Public D64NameA(), DiskHeaderA(), DiskIDA(), DemoNameA(), DemoStartA(), DirArtA() As String
    Public FileNameA(), FileAddrA(), FileOffsA(), FileLenA() As String
    Public tmpFileNameA(), tmpFileAddrA(), tmpFileOffsA(), tmpFileLenA() As String
    Public FileIOA() As Boolean
    Public tmpFileIOA() As Boolean
    Public BitsNeededForNextBundle As Integer = 0

    Public Prgs As New List(Of Byte())
    Public tmpPrgs As New List(Of Byte())

    Public VFileCnt As Integer = -1
    Public VFiles As New List(Of Byte())
    Public VFileNameA(), VFileAddrA(), VFileOffsA(), VFileLenA() As String
    Public VFileIOA() As Boolean
    Public tmpVFiles As New List(Of Byte())
    Public tmpVFileNameA(), tmpVFileAddrA(), tmpVFileOffsA(), tmpVFileLenA() As String
    Public tmpVFileIOA() As Boolean

    Public DiskNoA(), DFDiskNoA(), DFBundleNoA(), DiskBundleCntA(), DiskFileCntA() As Integer
    Public FilesInBundleA() As Integer
    Public PDiskNoA(), PSizeA() As Integer
    Public PNewBlockA() As Boolean
    Public FDiskNoA(), FBundleNoA(), FSizeA() As Integer
    Public TotalBundles As Integer = 0
    Public NewFile As String

    Public DiskSizeA() As Integer
    Public DiskStartBundle() As Integer
    Public FileSizeA() As Integer
    Public FBSDisk() As Integer
    Public BundleBytePtrA() As Integer
    Public BundleBitPtrA() As Integer
    Public BundleBitPosA() As Integer
    Public BundleSizeA() As Integer
    Public BundleBlockCntA() As Integer
    Public BundleOrigSizeA() As Integer
    Public UncompBundleSize As Double = 0

    Public bBuildDisk As Boolean = False

    Public SS, SE, LastSS, LastSE As Integer
    Public NewBundle As Boolean = False
    Public ScriptEntryType As String = ""
    Public ScriptEntry As String = ""
    Public ScriptLine As String = ""
    Public ScriptEntryArray() As String
    Public LastNonEmpty As Integer = -1

    Public LC(), NM, FM, LM As Integer
    Public SM1 As Integer = 0
    Public SM2 As Integer = 0
    Public OverMaxLit As Boolean = False

    Dim FirstFileOfDisk As Boolean = False
    Dim FirstFileStart As String = ""

    Public BlockPtr As Integer '= 255
    Public LastBlockCnt As Byte = 0
    Public LoaderBundles As Integer = 1
    Public FilesInBuffer As Byte = 1

    Public TmpSetNewBlock As Boolean = False
    Public SetNewBlock As Boolean = False      'This will fire at the previous bundle and will set NewBlock2
    Public NewBlock As Boolean = False     'This will fire at the specified bundle

    Public DirTrack, DirSector, DirPos As Integer
    Public DirArt As String = ""
    Public DirArtName As String = ""
    Private DirEntry As String = ""

    Private LastDirSector As Byte

    Public ScriptPath As String

    Public CmdLine As Boolean = False

    Private Loader() As Byte
    Private BlocksUsedBySaver As Integer = 0

    Public CompressBundleFromEditor As Boolean = False
    Public LastFileOfBundle As Boolean = False

    Public SaverSupportsIO As Boolean = False

    Private LoaderBlockCount As Byte

    Public Sub SetMaxSector()
        If DoOnErr Then On Error GoTo Err

        Select Case CT
            Case 1 To 17
                MaxSector = 20
        'LastSector = 17
            Case 18 To 24
                MaxSector = 18
        'LastSector = 15
            Case 25 To 30
                MaxSector = 17
        'LastSector = 15
            Case 31 To 40
                MaxSector = 16
                'LastSector = 13
        End Select

        Exit Sub
Err:
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Sub ResetArrays()
        If DoOnErr Then On Error GoTo Err

        DiskCnt = -1

        ReDim DiskNoA(DiskCnt), DiskBundleCntA(DiskCnt), DiskFileCntA(DiskCnt)
        ReDim D64NameA(DiskCnt), DiskHeaderA(DiskCnt), DiskIDA(DiskCnt), DemoNameA(DiskCnt), DemoStartA(DiskCnt), DirArtA(DiskCnt)
        ReDim DiskSizeA(DiskCnt)

        BundleCnt = -1
        ReDim PDiskNoA(BundleCnt), PSizeA(BundleCnt), BundleSizeA(BundleCnt), FilesInBundleA(BundleCnt), BundleOrigSizeA(BundleCnt)

        FileCnt = -1

        ReDim FileNameA(FileCnt), DFDiskNoA(FileCnt), DFBundleNoA(FileCnt), FileAddrA(FileCnt), FileOffsA(FileCnt), FileLenA(FileCnt)
        ReDim FDiskNoA(FileCnt), FBundleNoA(FileCnt), FSizeA(FileCnt)
        ReDim FileSizeA(FileCnt), FBSDisk(FileCnt)

        TotalBundles = 0

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")

    End Sub

    Public Function FindNextScriptEntry() As Boolean
        If DoOnErr Then On Error GoTo Err

        FindNextScriptEntry = True

NextLine:
        If Mid(Script, SS, 1) = Chr(13) Then                'Check if this is an empty line indicating a new section
            NewBundle = True
            SS += 2                                         'Skip vbCrLf
            SE = SS + 1
            GoTo NextLine
        ElseIf Mid(Script, SS, 1) = Chr(10) Then            'Line ends with vbLf
            NewBundle = True
            SS += 1                                         'Skip vbLf
            SE = SS + 1
            GoTo NextLine
        End If

NextChar:
        If (Mid(Script, SE, 1) <> Chr(13)) And (Mid(Script, SE, 1) <> Chr(10)) Then   'Look for vbCrLf and vbLF
            SE += 1                                         'Not EOL
            If SE <= Len(Script) Then                     'Go to next char if we haven't reached the end of the script
                GoTo NextChar
            Else
                'ScriptEntry = Strings.Mid(Script, SS, SE - SS)    'Reached end of script, finish this
```

```
entry
                'SS = SE + 2                                     'Skip EOL bytes
                'SE = SS + 1
                GoTo Done
            End If
        Else                                                'Found EOL
Done:       ScriptEntry = Mid(Script, SS, SE - SS)  'Finish this entry
            ScriptLine = ScriptEntry
            If Mid(Script, SE, 1) = Chr(13) Then            'Skip vbCrLf (2 chars)
                SS = SE + 2
            Else                                            'Otherwise skip 1 char onyl
                SS = SE + 1
            End If
            SE = SS + 1
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

        FindNextScriptEntry = False

    End Function

    Public Sub SplitEntry()
        If DoOnErr Then On Error GoTo Err

        LastNonEmpty = -1

        ReDim ScriptEntryArray(LastNonEmpty)

        ScriptEntryArray = Split(ScriptEntry, vbTab)

        'Remove empty strings (e.g. if there are to TABs between entries)
        For I As Integer = 0 To ScriptEntryArray.Length - 1
            If ScriptEntryArray(I) <> "" Then
                LastNonEmpty += 1
                ScriptEntryArray(LastNonEmpty) = ScriptEntryArray(I)
            End If
        Next

        If LastNonEmpty > -1 Then
            ReDim Preserve ScriptEntryArray(LastNonEmpty)
        Else
            ReDim ScriptEntryArray(0)
        End If

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Sub UpdateDiskSizeOnTheFly()
        If DoOnErr Then On Error GoTo Err

        CP = Track(18)

        If TracksPerDisk = ExtTracksPerDisk Then
            ReDim Preserve Disk(ExtBytesPerDisk - 1)
            BlocksFree = ExtSectorsPerDisk
            SectorsPerDisk = ExtSectorsPerDisk

            For Cnt As Integer = (36 + 7) * 4 To ((41 + 7) * 4) - 1
                Disk(Track(18) + Cnt) = 255
            Next
            For Cnt As Integer = 36 To 40
                Disk(Track(18) + ((Cnt + 7) * 4) + 0) = 17
                Disk(Track(18) + ((Cnt + 7) * 4) + 3) = 1
            Next
        Else
            ReDim Preserve Disk(StdBytesPerDisk - 1)
            BlocksFree = StdSectorsPerDisk
            SectorsPerDisk = StdSectorsPerDisk

            For Cnt As Integer = (36 + 7) * 4 To ((41 + 7) * 4) - 1
                Disk(Track(18) + Cnt) = 0
            Next
            For Cnt As Integer = 36 To 40
                Disk(Track(18) + ((Cnt + 7) * 4) + 0) = 0
                Disk(Track(18) + ((Cnt + 7) * 4) + 3) = 0
            Next
        End If

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Sub NewDisk()
        If DoOnErr Then On Error GoTo Err

        If TracksPerDisk = ExtTracksPerDisk Then
            ReDim Disk(ExtBytesPerDisk - 1)
            BlocksFree = ExtSectorsPerDisk
            SectorsPerDisk = ExtSectorsPerDisk
        Else
            ReDim Disk(StdBytesPerDisk - 1)
            BlocksFree = StdSectorsPerDisk
            SectorsPerDisk = StdSectorsPerDisk
        End If

        Dim B As Byte

        Dim Cnt As Integer

        CP = Track(18)
        Disk(CP) = &H12                                 'Track#18
        Disk(CP + 1) = &H1                              'Sector#1
        Disk(CP + 2) = &H41                             '"A"

        For Cnt = &H90 To &HAA                          'Name, ID, DOS type
            Disk(CP + Cnt) = &HA0
        Next

        '--------------------------------------------

        For Cnt = 1 To Len(DiskHeader)
            B = Ascii2Petscii(Asc(Mid(DiskHeader, Cnt, 1)))
            'If B > &H5F Then B -= &H20
            Disk(CP + &H8F + Cnt) = B
        Next

        '-------------------------------------------

        For Cnt = 1 To Len(DiskID)                  'SPRKL
            B = Ascii2Petscii(Asc(Mid(DiskID, Cnt, 1)))
            'If B > &H5F Then B -= &H20
            Disk(CP + &HA1 + Cnt) = B
        Next

        '-------------------------------------------

        For Cnt = 4 To (36 * 4) - 1
            Disk(CP + Cnt) = 255
        Next

        For Cnt = 1 To 17
            Disk(CP + (Cnt * 4) + 0) = 21
            Disk(CP + (Cnt * 4) + 3) = 31
        Next

        For Cnt = 18 To 24
            Disk(CP + (Cnt * 4) + 0) = 19
            Disk(CP + (Cnt * 4) + 3) = 7
        Next

        For Cnt = 25 To 30
            Disk(CP + (Cnt * 4) + 0) = 18
            Disk(CP + (Cnt * 4) + 3) = 3
        Next

        For Cnt = 31 To 35
            Disk(CP + (Cnt * 4) + 0) = 17
            Disk(CP + (Cnt * 4) + 3) = 1
        Next

        'If TracksPerDisk = ExtTracksPerDisk Then
        'For Cnt = (36 + 7) * 4 To ((41 + 7) * 4) - 1
        'Disk(CP + Cnt) = 255
        'Next
        'For Cnt = 36 To 40
        'Disk(CP + ((Cnt + 7) * 4) + 0) = 17
        'Disk(CP + ((Cnt + 7) * 4) + 3) = 1
        'Next
        'End If

        Disk(CP + (18 * 4) + 0) = 17
        Disk(CP + (18 * 4) + 1) = 252

        CT = 18
        CS = 1

        SetMaxSector()

        CP = Track(CT) + (256 * CS)
        Disk(CP + 1) = 255

        NextTrack = 1 : NextSector = 0

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Private Function SectorOK(T As Byte, S As Byte) As Boolean
        If DoOnErr Then On Error GoTo Err

        Dim BP As Integer   'BAM Position for Bit Change
        Dim BB As Integer   'BAM Bit

        BP = Track(18) + T * 4 + 1 + Int(S / 8)
        BB = 2 ^ (S Mod 8)    '=0-7
        If (Disk(BP) And BB) = 0 Then   'Block is already used
            SectorOK = False
        Else
            SectorOK = True                 'Block is unused
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

        SectorOK = False

    End Function

    Private Function FindNextFreeSector()
        If DoOnErr Then On Error GoTo Err

        FindNextFreeSector = True

        Dim Counter As Integer = 0
        Dim MaxS As Integer
CheckB:
        If SectorOK(CT, CS) = False Then
            CS += 1
            Counter += 1
            Select Case CT
                Case 1 To 17
                    MaxS = 21
                    If CS = 21 Then CS = 0
                Case 18 To 24
                    MaxS = 19
                    If CS = 19 Then CS = 0
                Case 25 To 30
                    MaxS = 18
                    If CS = 18 Then CS = 0
                Case 31 To 35
                    MaxS = 17
                    If CS = 17 Then CS = 0
            End Select
            If Counter < MaxS Then
                GoTo CheckB
            Else
                FindNextFreeSector = False
            End If
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")
        FindNextFreeSector = False

    End Function

    Public Sub DeleteBit(T As Byte, S As Byte, Optional UpdateFreeBlocks As Boolean = True)
        If DoOnErr Then On Error GoTo Err

        'Ignore tracks > 35
        'If T > 35 Then Exit Sub

        Dim BP As Integer   'BAM Position for Bit Change
        Dim BB As Integer   'BAM Bit

        BP = Track(18) + (T * 4) + 1 + Int(S / 8) + If(T > 35, 7 * 4, 0)
        BB = 255 - (2 ^ (S Mod 8))    '=0-7

        Disk(BP) = Disk(BP) And BB

        BP = Track(18) + (T * 4) + If(T > 35, 7 * 4, 0)
        Disk(BP) -= 1

        If UpdateFreeBlocks = True Then BlocksFree -= 1

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")

    End Sub

    Public Function AddInterleave(Optional IL As Byte = 5) As Boolean
        If DoOnErr Then On Error GoTo Err

        AddInterleave = True

        If TrackIsFull(CT) = True Then 'If this track is full, go to next and check again
            If (CT = 35) Or (CT = 18) Then         'Reached max track No, disk is full
                AddInterleave = False
                Exit Function
            End If
            CalcNextSector(IL)
            CT += 1

            If SystemFile = False Then
                If CT = 18 Then
                    CT = 19
                    CS = 3  'Need to skip 2 sectors while skipping Track 18 (the disk keeps spinning)
                End If
            End If
            'First sector in new track will be #1 and NOT #0!!!
            'CS = StartSector
        Else
            CalcNextSector(IL)
        End If

        AddInterleave = FindNextFreeSector()

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

        AddInterleave = False

    End Function

    Private Function TrackIsFull(T As Byte) As Boolean
        If DoOnErr Then On Error GoTo Err

        If Disk(Track(18) + T * 4) = 0 Then
            TrackIsFull = True
        Else
            TrackIsFull = False
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Function

    Private Sub CalcNextSector(Optional IL As Byte = 5)
        If DoOnErr Then On Error GoTo Err

        Select Case CT

            Case 1 To 17    '21 sectors, 0-20

                If CS > 20 Then
                    CS -= 21
                    If CS > 0 Then CS -= 1
                End If
                CS += IL     'IL=4 always
                If CS > 20 Then
                    CS -= 21
                    If CS > 0 Then CS -= 1
                End If

            Case 18         'Handle Dir Track separately
                If CS > 18 Then CS -= 19
                CS += IL
                If CS > 18 Then CS -= 19

            Case 19 To 24   '19 sectors, 0-18
                If CS > 18 Then CS -= 19
                CS += IL     'IL=3 always
                If CS > 18 Then CS -= 19

            Case 25 To 30   '18 sectors, 0-17
                If CS > 17 Then CS -= 18
                CS += IL     'IL=3 always
                If CS > 17 Then CS -= 18

            Case 31 To 35   '17 sectors, 0-16
                If CS > 16 Then CS -= 17
                CS += IL     'IL=3 always
                If CS > 16 Then CS -= 17

        End Select

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Function EORtransform(Input As Byte) As Byte

        EORtransform = Input

        Select Case (Input And &H9)
            Case 0, 9
                Return EORtransform Xor &H7F
            Case 1, 8
                Return EORtransform Xor &H76
        End Select

        'Select Case (Input And &H99)
        'Case 0, 9, &H90, &H99
        'Return EORtransform Xor &HFF
        'Case 1, 8, &H91, &H98
        'Return EORtransform Xor &HF6
        'Case &H10, &H19, &H80, &H89
        'Return EORtransform Xor &H6F
        'Case &H11, &H18, &H81, &H88
        'Return EORtransform Xor &H66
        'End Select

        'Select Case (Input And &H69)
        'Case 0, 9, &H60, &H69
        'Return EORtransform Xor &HFF
        'Case 1, 8, &H61, &H68
        'Return EORtransform Xor &HF6
        'Case &H20, &H29, &H40, &H49
        'Return EORtransform Xor &H9F
        'Case &H21, &H28, &H41, &H48
        'Return EORtransform Xor &H96
        'End Select


    End Function

    Public Sub InjectDirBlocks()

        'DirBlocks(0) = EORtransform(Track)
        'DirBlocks(1) = EORtransform(Sector)
        'DirBlocks(2) = EORtransform(Remaining sectors on track)
        'DirBlocks(3) = BitPtr

        If BundleNo >= 0 Then     'This may be unneccesary, there is always at least 1 bundle on the
disk
            For I As Integer = BundleNo + 1 To 127
```

```
                DirBlocks((I * 4) + 3) = DirBlocks((BundleNo * 4) + 3)
                DirPtr(I) = DirPtr(BundleNo)
            Next
        End If

        For I As Integer = 0 To 127
            DirBlocks(I * 4) = EORtransform(TabT(DirPtr(I)))
            DirBlocks((I * 4) + 1) = EORtransform(TabStartS(TabT(DirPtr(I))))
            DirBlocks((I * 4) + 2) = EORtransform(TabSCnt(DirPtr(I)))
        Next

        'Resort directory sectors to allow simple copy from $0100 to $0700
        'Dir Block: $00,$ff,$fe,$fd,$fc,...,$01
        'Buffer:    $00,$01,$02,$03,$04,...,$ff

        Dim DB0(255), DB1(255) As Byte
        Dim B As Integer = 0         '255 for SD2
        For I As Integer = 0 To 255
            DB0(B) = DirBlocks(I)
            DB1(B) = DirBlocks(I + 256)
            B -= 1
            If B < 0 Then B += 256
        Next

        For I = 0 To 255
            DirBlocks(I) = DB0(I)
            DirBlocks(I + 256) = DB1(I)
        Next

        For I As Integer = 0 To 511
            Disk(Track(18) + (17 * 256) + I) = DirBlocks(I)
        Next

    End Sub

    Public Function InjectDriveCode(idcDiskID As Byte, idcFileCnt As Byte, idcNextID As Byte, Optional
TestDisk As Boolean = False) As Boolean
        If DoOnErr Then On Error GoTo Err

        InjectDriveCode = True

        Dim I, Cnt As Integer

        'If TestDisk = True Then
        'Drive = My.Resources.SDT
        'Else
        Drive = My.Resources.SD
        ReDim Preserve Drive((6 * 256) + 1)
        'End If

        Dim B3(255) As Byte
        Dim B As Integer = 0         '255 for SD2

        'Resort and EOR transform Block 3
        For I = 0 To 255
            B3(B) = EORtransform(Drive((3 * 256) + I + 2))
            B -= 1
            If B < 0 Then B += 256
        Next
        '-------------------
        '   VersionInfo
        '-------------------
        'Add version info: YY MM DD VV
        Dim VI As Integer = &H5B
        Drive(VI + 0 + 2) = (Int(My.Application.Info.Version.Build / 10) * 16) +
(My.Application.Info.Version.Build Mod 10)
        Drive(VI + 1 + 2) = (Int(My.Application.Info.Version.Revision / 1000) * 16) +
(Int(My.Application.Info.Version.Revision / 100) Mod 10)
        Drive(VI + 2 + 2) = ((Int(My.Application.Info.Version.Revision / 10) Mod 10) * 16) +
(My.Application.Info.Version.Revision Mod 10)
        Drive(VI + 4 + 2) = (My.Application.Info.Version.Major * 16) +
My.Application.Info.Version.Minor

        '-------------------
        '   ProductID
        '-------------------
        'Add Product ID (add 2 to address for PRG header)
        Dim PID As Integer = &H1B
        Drive(PID + 0 + 2) = Int(ProductID / &H10000) And &HFF
        Drive(PID + 1 + 2) = Int(ProductID / &H100) And &HFF
        Drive(PID + 2 + 2) = ProductID And &HFF

        '-------------------
        '   NoFlipTab
        '-------------------
        'Save last, "dummy" bundle info, needs REVERSED EOR Transform as it is used in the drive code
(add 2 to address for PRG header)
        'Dim NFT As Integer = &H21
        'Drive(NFT + 0 + 2) = TabT(LastBufferCnt)
        'Drive(NFT + 1 + 2) = TabStartS(TabT(LastBufferCnt))
        'Drive(NFT + 2 + 2) = TabSCnt(LastBufferCnt)
        'Drive(NFT + 3 + 2) = EORtransform(LastBitPtr)

        'Resort blocks in drive code:
        For I = 0 To 255
            Drive((3 * 256) + I + 2) = Drive((4 * 256) + I + 2)     'Copy ZP GCR Tab and GCR loop to
block 3 for loading
            Drive((4 * 256) + I + 2) = Drive((5 * 256) + I + 2)     'Copy Init code to block 4 for
loading
            Drive((5 * 256) + I + 2) = B3(I)                        'Copy original block 3 EOR
transformed to block 5 to be loaded by init code
        Next

        CT = 18
        CS = 11
        For Cnt = 0 To 5        '6 blocks to be saved: 18:11, 18:12, 18:13, 18:14, 18:15, (18:16 -
block 5)
            For I = 0 To 255
                If Drive.Length > Cnt * 256 + I + 2 Then
                    Disk(Track(CT) + CS * 256 + I) = Drive(Cnt * 256 + I + 2)
                End If
            Next
            DeleteBit(CT, CS, False)
            CS += 1
        Next

        'Next Side Info on last 2 bytes of BAM!!! (Buffer address: $0101-$0102)
        Disk(Track(18) + (0 * 256) + 255) = EORtransform(idcDiskID)
        Disk(Track(18) + (0 * 256) + 251) = EORtransform(idcNextID)

        'Add Custom Interleave Info (Buffer address: $0103-$0107)
        Disk(Track(18) + (0 * 256) + 254) = EORtransform(256 - IL3)
        Disk(Track(18) + (0 * 256) + 253) = EORtransform(256 - IL2)
        Disk(Track(18) + (0 * 256) + 252) = EORtransform(256 - IL1)
        'Disk(Track(18) + (0 * 256) + 250) = EORtransform(IL0)
        Disk(Track(18) + (0 * 256) + 250) = EORtransform(256 - IL0)

        '"Dummy" bundle info EOR transformed - to be copied to NoFlipTab after disk flip (Buffer address: $0108-$010b)
        'Disk(Track(18) + (0 * 256) + 248) = EORtransform(TabT(LastBufferCnt))
        'Disk(Track(18) + (0 * 256) + 247) = EORtransform(TabStartS(TabT(LastBufferCnt)))
        'Disk(Track(18) + (0 * 256) + 246) = EORtransform(TabSCnt(LastBufferCnt))
        'Disk(Track(18) + (0 * 256) + 245) = LastBitPtr

        'Add IncludeSaveCode flag (Buffer address: $010c)
        BlocksUsedBySaver = 0
        If bSaverPlugin Then
            Disk(Track(18) + (0 * 256) + 249) = EORtransform(2)
            InjectSaverPlugin()
        Else
            Disk(Track(18) + (0 * 256) + 249) = EORtransform(0)
        End If

        'Also add Product ID to BAM, EOR-transformed (Buffer address: $010d-$010f)
        Disk(Track(18) + (0 * 256) + 248) = EORtransform(Int(ProductID / &H10000) And &HFF)
        Disk(Track(18) + (0 * 256) + 247) = EORtransform(Int(ProductID / &H100) And &HFF)
        Disk(Track(18) + (0 * 256) + 246) = EORtransform(ProductID And &HFF)

        'Add NextID and IL0-IL3 to ZPTab (could be done before Drive code is injected)
        Dim ZPNextIDLoc As Integer = &H60

        Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 0) = 256 - IL3
        Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 1) = 256 - IL2
        Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 2) = 256 - IL1
        Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 3) = idcNextID
        Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 4) = 256 - IL0

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")
        Resume
        InjectDriveCode = False

    End Function

    Private Sub InjectSaverPlugin()
        If DoOnErr Then On Error GoTo Err

        If bSaverPlugin = False Then Exit Sub
        If HSFile.Length = 0 Then Exit Sub
        If HSFileName = "" Then Exit Sub
        If BundleNo > 125 Then
            MsgBox("The Hi-Score File Saver Plugin cannot be added to the disk because the number of file bundles exceeds 126!" + vbNewLine + vbNewLine +
            "The Plugin and the Hi-Score File would use bundle indices $7e and $7f, respectively.", vbOKOnly + vbExclamation, "Hi-Score File Saver Plugin Error")
            Exit Sub
        End If

        BlocksUsedBySaver = Int(HSLength / &H100) + 1 + 2

        If BlocksFree < BlocksUsedBySaver Then
            MsgBox("The Hi-Score File Saver Plugin cannot be added because there is not enough free space on the disk!" + vbNewLine + vbNewLine +
            "The Plugin and the Hi-Score File would need " + BlocksUsedBySaver.ToString + " free blocks but there " + If(BlocksFree = 1, "is", "are") + " only " + BlocksFree.ToString +
            " block" + If(BlocksFree = 1, "", "s") + " available on the disk.", vbOKOnly +
vbExclamation, "Hi-Score File Saver Plugin Error")
            Exit Sub
        End If

        If InStr(HSFileName, "*") <> 0 Then
            SaverSupportsIO = True
            Replace(HSFileName, "*", "")
        Else
            SaverSupportsIO = False
        End If

        Dim SaveCode() As Byte

        If SaverSupportsIO Then
            SaveCode = My.Resources.SSIO
        Else
            SaveCode = My.Resources.SS
        End If

        'UpdateZP BUG REPORTED BY Rico/Pretzel Logic
        'WE ALSO NEED TO UPDATE ZP OFFSET IN THE SAVER CODE!!!

        'Convert LoaderZP to byte - it has already been validated in UpdateZP
        Dim ZP As Byte = Convert.ToByte(LoaderZP, 16)

        If ZP <> 2 Then
            Dim OPC_STAZP As Byte = &H85    'ZP, ZP+1, Bits
            Dim OPC_LDAZPY As Byte = &HB1   'ZP

            Dim ZPBase As Byte = &H2

            For I As Integer = 0 To 249
                If (SaveCode(I) = OPC_STAZP) Or (SaveCode(I) = OPC_LDAZPY) Then
                    If SaveCode(I + 1) = ZPBase Then
                        SaveCode(I + 1) = ZP
                        I += 1
                    End If
                End If
            Next

            For I As Integer = 0 To 249
                If (SaveCode(I) = OPC_STAZP) And (SaveCode(I + 1) = ZPBase + 1) Then
                    SaveCode(I + 1) = ZP + 1
                    I += 1
                End If
            Next

            For I As Integer = 0 To 249
                If (SaveCode(I) = OPC_STAZP) And (SaveCode(I + 1) = ZPBase + 2) Then
                    SaveCode(I + 1) = ZP + 2
                    I += 1
                End If
            Next

        End If

        SaveCode(2 + &H3) = Int(HSLength / &H100) + 1     'Add 2 for PRG offset!
        SaveCode(2 + &H13) = (HSAddress - 1) And &HFF
        SaveCode(2 + &H1A) = Int((HSAddress - 1) / &H100)

        'For I As Integer = 0 To SaveCode.Count - 3
        ''Find JSR $01e5 (JSR Set01 - expected to remain constant)
        'If SaveCode(I) = &H20 And SaveCode(I + 1) = &HE5 And SaveCode(I + 2) = &H1 Then
        'SaveCode(I - 11) = (HSAddress - 1) And &HFF
        'SaveCode(I - 4) = Int((HSAddress - 1) / &H100)
        'Exit For
        'End If
        'Next

        'Calculate sector pointer on disk
        Dim SctPtr As Integer = SectorsPerDisk - 2 - (Int(HSLength / 256) + 1)

        'Identify first T/S of the saver plugin
        CT = TabT(SctPtr)
        CS = TabS(SctPtr)

        'Copy first block of saver plugin to disk
        For I As Integer = 0 To 255
            Disk(Track(CT) + (CS * 256) + I) = SaveCode(2 + I)
        Next

        'Mark sector off in BAM
        DeleteBit(CT, CS, True)

        'Add plugin to directory
        Disk(Track(18) + (18 * 256) + 8) = EORtransform(CT)                'DirBlocks(0) = EORtransform(Track) = 35
        Disk(Track(18) + (18 * 256) + 7) = EORtransform(TabStartS(CT))    'DirBlocks(1) = EORtransform(Sector) = First sector of Track(35) (not first sector of file!!!)
        Disk(Track(18) + (18 * 256) + 6) = EORtransform(TabSCnt(SctPtr))  'DirBlocks(2) = EORtransform(Remaining sectors on track)
        Disk(Track(18) + (18 * 256) + 5) = &HFE                            'DirBlocks(3) = BitPtr

        'Next Sector
        SctPtr += 1

        'Second T/S of saver plugin
        CT = TabT(SctPtr)
        CS = TabS(SctPtr)

        'Copy second block of saver plugin to disk
        For I As Integer = 0 To SaveCode.Length - 256 - 1 - 2
            Dim J As Integer = 0 - I
            If J < 0 Then J += 256
            Disk(Track(CT) + (CS * 256) + J) = EORtransform(SaveCode(256 + 2 + I))
        Next

        'Mark sector off in BAM
        DeleteBit(CT, CS, True)

        'Add SaveFile
        SctPtr += 1

        CT = TabT(SctPtr)
        CS = TabS(SctPtr)

        Disk(Track(18) + (18 * 256) + 4) = EORtransform(CT)                  'DirBlocks(0) = EORtransform(Track) = 35
        Disk(Track(18) + (18 * 256) + 3) = EORtransform(TabStartS(CT))      'DirBlocks(1) = EORtransform(Sector) = First sector of Track(35) (not first sector of file!!!)
        Disk(Track(18) + (18 * 256) + 2) = EORtransform(TabSCnt(SctPtr))    'DirBlocks(2) = EORtransform(Remaining sectors on track)
        Disk(Track(18) + (18 * 256) + 1) = &HFE                              'DirBlocks(3) = BitPtr

        DeleteBit(CT, CS, True)

        Dim Buffer(255) As Byte
        Dim HSStartAdd As Integer = HSAddress + HSLength - 1
        Dim BlockCnt = Int(HSLength / 256)

        'First block
        Buffer(0) = 0
        Buffer(1) = EORtransform(Int(HSLength / 256))               'Remaining block count (EOR 
transformed)
        Buffer(255) = &HFE                                          'First byte of block
        Buffer(254) = &H81                                          'Bit stream
        Buffer(253) = HSStartAdd Mod 256                            'Last byte's address (Lo)
        If SaverSupportsIO Then
            Buffer(252) = 0                                         'I/O flag
            Buffer(251) = Int(HSStartAdd / 256)                     'Last byte's address (Hi)
            Buffer(250) = 0                                         'LongLit flag
            Buffer(249) = &HF6                                      'Number of literals - 1
        Else
            Buffer(252) = Int(HSStartAdd / 256)                     'Last byte's address (Hi)
            Buffer(251) = 0                                         'LongLit flag
            Buffer(250) = &HF7                                      'Number of literals - 1
        End If

        For I As Integer = 2 To If(SaverSupportsIO, 248, 249)
            Buffer(I) = HSFile(HSLength - 1 - If(SaverSupportsIO, 248, 249) + I)
        Next

        For I As Integer = 0 To 255
            Disk(Track(CT) + CS * 256 + I) = Buffer(I)
        Next

        If SaverSupportsIO Then
            HSStartAdd -= &HF7
            HSLength -= &HF7
        Else
            HSStartAdd -= &HF8
            HSLength -= &HF8
        End If

        'Blocks 1 to BlockCnt-1
        For I As Integer = 1 To BlockCnt - 1

            SctPtr += 1

            CT = TabT(SctPtr)
            CS = TabS(SctPtr)

            DeleteBit(CT, CS, True)

            ReDim Buffer(255)

            Buffer(0) = &H81                                    'Bit stream
            Buffer(255) = HSStartAdd Mod 256                    'Last byte's address (Lo)
            If SaverSupportsIO Then
                Buffer(254) = 0                                     'I/O flag
                Buffer(253) = Int(HSStartAdd / 256)                 'Last byte's address (hi)
                Buffer(252) = 0                                     'LongLit flag
                Buffer(251) = &HF9                                  'Number of literals - 1
            Else
                Buffer(254) = Int(HSStartAdd / 256)                 'Last byte's address (hi)
                Buffer(253) = 0                                     'LongLit flag
                Buffer(252) = &HFA                                  'Number of literals - 1
            End If

            For J As Integer = 1 To If(SaverSupportsIO, 250, 251)
                Buffer(J) = HSFile(HSLength - 1 - If(SaverSupportsIO, 250, 251) + J)
            Next

            For J As Integer = 0 To 255
                Disk(Track(CT) + (CS * 256) + J) = Buffer(J)
            Next

            If SaverSupportsIO Then
                HSStartAdd -= &HFA
                HSLength -= &HFA
            Else
                HSStartAdd -= &HFB
                HSLength -= &HFB
            End If

        Next

        'Last block of Hi-Score File
        SctPtr += 1

        CT = TabT(SctPtr)
        CS = TabS(SctPtr)

        DeleteBit(CT, CS, True)

        ReDim Buffer(255)

        Buffer(0) = &H81                                        'Bit stream
        Buffer(1) = EORtransform(0)                             'New block count = 0 (eor transformed)
        Buffer(255) = HSStartAdd Mod 256                        'Last byte's address (Lo)
        If SaverSupportsIO Then
            Buffer(254) = 0                                     'I/O flag
            Buffer(253) = Int(HSStartAdd / 256)                 'Last byte's address (Hi)
            Buffer(252) = 0                                     'LongLit flag
            Buffer(251) = HSLength - 1                          'Number of remaining literals - 1
        Else
            Buffer(254) = Int(HSStartAdd / 256)                 'Last byte's address (Hi)
            Buffer(253) = 0                                     'LongLit flag
            Buffer(252) = HSLength - 1                          'Number of remaining literals - 1
        End If

        For I As Integer = 0 To HSLength - 1
            Buffer(If(SaverSupportsIO, 251, 252) - HSLength + I) = HSFile(I)
        Next

        Buffer(If(SaverSupportsIO, 251, 252) - HSLength - 1) = &HF8                       'End of File Bundle flag

        For I As Integer = 0 To 255
            Disk(Track(CT) + (CS * 256) + I) = Buffer(I)
        Next

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")

```

```
    End Sub

    Public Function InjectLoader(DiskIndex As Integer, T As Byte, S As Byte, IL As Byte, Optional TestDisk As Boolean = False) As Boolean
        If DoOnErr Then On Error GoTo Err

        InjectLoader = True

        Dim B, I, Cnt, W As Integer
        Dim ST, SS, A, AdLo, AdHi As Byte

        'Check if we have a Demo Start Address
        If DiskIndex > -1 Then
            If DemoStartA(DiskIndex) <> "" Then B = Convert.ToInt32(DemoStartA(DiskIndex), 16)
        Else
            If DemoStart <> "" Then B = Convert.ToInt32(DemoStart, 16)
        End If

        'No Demo Start Address, check if we have the first file's start address
        If B = 0 Then
            If FirstFileStart <> "" Then B = Convert.ToInt32(FirstFileStart, 16)
        End If

        If B = 0 Then
            MsgBox("Unable to build demo disk." + vbNewLine + vbNewLine + "Missing start address", vbOKOnly)
            InjectLoader = False
            Exit Function
        End If

        AdLo = (B - 1) Mod 256
        AdHi = Int((B - 1) / 256)

        If TestDisk = False Then
            Loader = My.Resources.SL
            UpdateZP()
        Else
            Loader = My.Resources.SLT
        End If

        For I = 0 To Loader.Length - 6      'Find JMP Sparkle_LoadFetched instruction
            If (Loader(I) = &H10) And (Loader(I + 3) = &HAD) And (Loader(I + 5) = &H4C) Then
                Loader(I) = AdHi            'Hi Byte return address at the end of Loader
                Loader(I + 3) = AdLo        'Lo Byte return address at the end of Loader
                Exit For
            End If
        Next

        'Number of blocks in Loader
        LoaderBlockCount = Int(Loader.Length / 254)
        If (Loader.Length) Mod 254 <> 0 Then
            LoaderBlockCount += 1
        End If

        CT = T
        CS = S

        For I = 0 To LoaderBlockCount - 1
            ST = CT
            SS = CS
            For Cnt = 0 To 253
                If (I * 254) + Cnt < Loader.Length Then
```

```
                    Disk(Track(CT) + (CS * 256) + 2 + Cnt) = Loader((I * 254) + Cnt)
                End If
            Next
            DeleteBit(CT, CS, False)

            AddInterleave(IL)   'Go to next free sector with Interleave IL
            If I < LoaderBlockCount - 1 Then
                Disk(Track(ST) + (SS * 256) + 0) = CT
                Disk(Track(ST) + (SS * 256) + 1) = CS
            Else
                Disk(Track(ST) + (SS * 256) + 0) = 0
                If Loader.Length Mod 254 = 0 Then
                    Disk(Track(ST) + (SS * 256) + 1) = 254 + 1
                Else
                    Disk(Track(ST) + (SS * 256) + 1) = ((Loader.Length) Mod 254) + 1
                End If
            End If
        Next

        'AddDemoNameToDisk(DiskIndex, T, S)

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

        InjectLoader = False

    End Function

    Private Sub AddDemoNameToDisk(DiskIndex As Integer, T As Byte, S As Byte)
        If DoOnErr Then On Error GoTo Err

        Dim B, Cnt As Integer
        Dim DN As String = ""
        Dim A As Byte

        If DiskIndex > -1 Then
            DN = If(DemoNameA(DiskIndex) <> "", DemoNameA(DiskIndex), "")
        Else
            DN = If(DemoName <> "", DemoName, "")
        End If

        If DN = "" Then
            'No DemoName defined, check if we have a DirArt file attached
            If DirArtName <> "" Then
                'Dirart attached, we will add first dir entry there
                Exit Sub
                'Else
                'No DirArt - we need a default dir entry
                'DemoName = DefaultDemoName
            End If
        End If

        CT = 18 : CS = 1
        Cnt = Track(CT) + (CS * 256)
SeekNewDirBlock:
        If Disk(Cnt) <> 0 Then
            Cnt = Track(Disk(Cnt)) + Disk(Cnt + 1) * 256
            GoTo SeekNewDirBlock
        Else
```

```
            B = 2
SeekNewEntry:
            If Disk(Cnt + B) = &H0 Then

                Disk(Cnt + B) = &H82
                Disk(Cnt + B + 1) = T
                Disk(Cnt + B + 2) = S
                For W = 0 To 15
                    Disk(Cnt + B + 3 + W) = &HA0
                Next

                For W = 0 To Len(DN) - 1
                    A = Ascii2Petscii(Asc(Mid(DN, W + 1, 1)))
                    'If A > &H5F Then A -= &H20
                    Disk(Cnt + B + 3 + W) = A
                Next
                Disk(Cnt + B + &H1C) = LoaderBlockCount    'Length of boot loader in blocks
            Else
                B += 32
                If B < 256 Then
                    GoTo SeekNewEntry
                Else
                    CS += 4
                    If CS > 18 Then S -= 18
                    Disk(Cnt) = CT
                    Disk(Cnt + 1) = CS
                    Cnt = Track(CT) + CS * 256
                    Disk(Cnt) = 0
                    Disk(Cnt + 1) = 255
                    GoTo SeekNewDirBlock
                End If
            End If
        End If

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub
    Private Sub UpdateZP()
        If DoOnErr Then On Error GoTo Err

        'Check string length
        If LoaderZP.Length < 2 Then
            LoaderZP = Left("02", 2 - LoaderZP.Length) + LoaderZP
        ElseIf LoaderZP.Length > 2 Then
            LoaderZP = Right(LoaderZP, 2)
        End If

        'Convert LoaderZP to byte
        Dim ZP As Byte = Convert.ToByte(LoaderZP, 16)

        'ZP cannot be $00, $01, or $ff
        If ZP < 2 Then
            MsgBox("Zeropage value cannot be less than $02." + vbNewLine + vbNewLine + "ZP is
corrected to $02. Please update the ZP entry in your script!", vbInformation + vbOKOnly)
            ZP = 2
            LoaderZP = "02"
        End If
        If ZP > &HFD Then
```

```
            MsgBox("Zeropage value cannot be greater than $fd." + vbNewLine + vbNewLine + "ZP is corrected to $fd. Please update the ZP entry in your script!", vbInformation + vbOKOnly)
            ZP = &HFD
            LoaderZP = "fd"
        End If

        'ZP=02 is the default, no need to update
        If ZP = 2 Then Exit Sub

        'ZPUpdate BUG REPORTED BY Rico/Pretzel Logic

        'Find the JMP $0700 sequence in the code to identify the beginning of loader
        Dim LoaderBase As Integer = &HFFFF
        For I As Integer = 0 To Loader.Length - 1 - 2
            If (Loader(I) = &H4C) AndAlso (Loader(I + 1) = &H0) AndAlso (Loader(I + 2) = &H7) Then
                LoaderBase = I + 3
                Exit For
            End If
        Next

        If LoaderBase = &HFFFF Then
            MsgBox("Zeropage offset could not updated. Sparkle will use the default zeropage offset value of $02.", vbInformation + vbOKOnly, "Error updating zeropage offset")
            Exit Sub
        End If

        Dim OPC_STAZP As Byte = &H85
        Dim OPC_ADCZP As Byte = &H65
        Dim OPC_STAZPY As Byte = &H91
        Dim OPC_DECZP As Byte = &HC6
        Dim OPC_LDAZP As Byte = &HA5
        Dim OPC_RORZP As Byte = &H66
        Dim OPC_ASLZP As Byte = &H6
        Dim OPC_EORIMM As Byte = &H49
        'Dim OPC_LDAZPY As Byte = &HB1

        Dim ZPBase As Byte = &H2

        For I As Integer = LoaderBase To Loader.Length - 1 - 1
            If (Loader(I) = OPC_STAZP) Or
               (Loader(I) = OPC_ADCZP) Or
               (Loader(I) = OPC_STAZPY) Then

                If (Loader(I + 1) = ZPBase) And (Loader(I + 2) <> OPC_EORIMM) Then  'Skip STA $0265 EOR #$FF
                    Loader(I + 1) = ZP
                    I += 1
                End If
            End If
        Next

        For I As Integer = LoaderBase To Loader.Length - 1 - 1
            If (Loader(I) = OPC_STAZP) Or
               (Loader(I) = OPC_DECZP) Or
               (Loader(I) = OPC_LDAZP) Then

                If Loader(I + 1) = ZPBase + 1 Then
                    Loader(I + 1) = ZP + 1
                    I += 1
                End If
            End If
        Next

        For I As Integer = LoaderBase To Loader.Length - 1 - 1
            If (Loader(I) = OPC_STAZP) Or
               (Loader(I) = OPC_RORZP) Or
               (Loader(I) = OPC_ASLZP) Then

                If Loader(I + 1) = ZPBase + 2 Then
                    Loader(I + 1) = ZP + 2
                    I += 1
                End If
            End If
        Next

        ''ZP                                     Instructions       Types
        'Loader(LoaderBase + &HAF) = ZP      'STA ZP                STA ZP
        'Loader(LoaderBase + &HD9) = ZP      'ADC ZP                            ADC ZP
        'Loader(LoaderBase + &HDB) = ZP      'STA ZP                STA (ZP),Y
        'Loader(LoaderBase + &HEE) = ZP      'ADC ZP
        'Loader(LoaderBase + &HF0) = ZP      'STA ZP
        'Loader(LoaderBase + &HFF) = ZP      'STA (ZP),Y
        'Loader(LoaderBase + &H10E) = ZP     'ADC ZP
        'Loader(LoaderBase + &H110) = ZP     'STA ZP
        'Loader(LoaderBase + &H11C) = ZP     'ADC ZP
        'Loader(LoaderBase + &H12D) = ZP     'STA (ZP),Y
        ''ZP+1
        'Loader(LoaderBase + &HBE) = ZP + 1  'STA ZP+1           STA ZP+1
        'Loader(LoaderBase + &HE3) = ZP + 1  'DEC ZP+1           DEC ZP+1
        'Loader(LoaderBase + &HF4) = ZP + 1  'DEC ZP+1           LDA ZP+1
        'Loader(LoaderBase + &H114) = ZP + 1 'DEC ZP+1
        'Loader(LoaderBase + &H121) = ZP + 1 'LDA ZP+1
        ''Bits
        'Loader(LoaderBase + &H1) = ZP + 2   'STA Bits           STA Bits
        'Loader(LoaderBase + &H1C) = ZP + 2  'ROR Bits           ROR Bits
        'Loader(LoaderBase + &HA4) = ZP + 2  'STA Bits           ASL Bits
        'Loader(LoaderBase + &H132) = ZP + 2 'ASL Bits           ROL Bits
        'Loader(LoaderBase + &H13D) = ZP + 2 'STA Bits
        'Loader(LoaderBase + &H141) = ZP + 2 'ROL Bits
        'Loader(LoaderBase + &H14C) = ZP + 2 'STA Bits

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Function ConvertIntToHex(HInt As Integer, SLen As Integer) As String
        If DoOnErr Then On Error GoTo Err

        ConvertIntToHex = LCase(Hex(HInt))

        If Len(ConvertIntToHex) < SLen Then
            ConvertIntToHex = Left(StrDup(SLen, "0"), SLen - ConvertIntToHex.Length) + ConvertIntToHex
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

```

```vb
        ConvertIntToHex = ""

    End Function

    Public Function UpdateBAM(DiskIndex As Integer) As Boolean
        If DoOnErr Then On Error GoTo Err

        UpdateBAM = True

        Dim Cnt As Integer
        Dim B As Byte
        CP = Track(18)

        For Cnt = &H90 To &HAA  'Name, ID, DOS type
            Disk(CP + Cnt) = &HA0
        Next

        If DiskHeaderA(DiskIndex) = "" Then DiskHeaderA(DiskIndex) = "demo disk" + If(DiskCnt > 0, " " + (DiskIndex + 1).ToString, "")

        '--------------------------------------------

        For Cnt = 1 To Len(DiskHeaderA(DiskIndex))
            B = Ascii2Petscii(Asc(Mid(DiskHeaderA(DiskIndex), Cnt, 1)))
            'If B > &H5F Then B -= &H20
            Disk(CP + &H8F + Cnt) = B
        Next

        '--------------------------------------------

        If DiskIDA(DiskIndex) <> "" Then
            For Cnt = 1 To Len(DiskIDA(DiskIndex))
                B = Ascii2Petscii(Asc(Mid(DiskIDA(DiskIndex), Cnt, 1)))
                'If B > &H5F Then B -= &H20
                Disk(CP + &HA1 + Cnt) = B
            Next
        Else
            For Cnt = 1 To Len(DiskID)
                B = Ascii2Petscii(Asc(Mid(DiskID, Cnt, 1)))
                'If B > &H5F Then B -= &H20
                Disk(CP + &HA1 + Cnt) = B
            Next
        End If

        '--------------------------------------------

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")

        UpdateBAM = False

    End Function

    Public Sub MakeTestDisk()
        If DoOnErr Then On Error GoTo Err

        Dim B As Byte
        Dim TDiff As Integer = Track(19) - Track(18)
        Dim SMax As Integer

        DemoStart = "0820"
        DemoName = "sparkle test"
        DiskHeader = "spakle test"
        DiskID = " 2019"

        NewDisk()

        For T As Integer = 1 To 35
            Select Case T
                Case 1 To 17
                    SMax = 20
                Case 19 To 24
                    SMax = 18
                Case 25 To 30
                    SMax = 17
                Case 31 To 35
                    SMax = 16
            End Select
            If T <> 18 Then
                For S As Integer = 0 To SMax
                    'For I As Integer = 0 To 255
                    'Select Case (I And 15)
                    'Case 0, 2, 4, 6, 9, 11, 13, 15
                    'B = (I And 15) Xor &HF
                    'Case Else
                    'B = (I And 15) Xor &H6
                    'End Select
                    '
                    'Select Case Int(I / 16)
                    'Case 0, 2, 4, 6, 9, 11, 13, 15
                    'B += (I And &HF0) Xor &HF0
                    'Case Else
                    'B += (I And &HF0) Xor &H60
                    'End Select
                    'Disk(Track(T) + S * 256 + I) = I
                    'Next
                    For I As Integer = S To S + 255
                        B = I Mod 256
                        Disk(Track(T) + S * 256 + I - S) = B
                    Next
                    DeleteBit(T, S, True)
                Next
            End If
        Next

        InjectLoader(-1, 18, 7, 1, True)
        InjectDriveCode(1, 255, 1, True)

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Function BuildDemoFromScript(Optional SaveIt As Boolean = True) As Boolean
        If DoOnErr Then On Error GoTo Err

        TotLits = 0
        TotSM = 0
        TotNMM = 0
        TotFMM = 0
        TotNLM = 0
        TotFLM = 0

        BuildDemoFromScript = True

        'Generate Product ID unique to this build - it will be the same for all disks in this build
        Randomize()
        ProductID = Int(Rnd() * &HFFFFFF)

        TotLit = 0 : TotMatch = 0

        SS = 1 : SE = 1

        'Check if this is a valid script
        If FindNextScriptEntry() = False Then GoTo NoDisk

        If ScriptEntry <> ScriptHeader Then
            MsgBox("Invalid Loader Script file!", vbExclamation + vbOKOnly)
            GoTo NoDisk
        End If

        TotalBits = 0

        CurrentBundle = 0
        DiskCnt = -1
        TotalBundles = 0
        'DiskLoop = 0    'Reset Loop variable
        'Reset Disk Variables
        If ResetDiskVariables() = False Then GoTo NoDisk
        Dim NewD As Boolean = True
        NewBundle = False
        TmpSetNewBlock = False

FindNext:
        LastSS = SS
        LastSE = SE
        If FindNextScriptEntry() = False Then GoTo NoDisk
        'Split String
        If SplitScriptEntry() = False Then GoTo NoDisk
        'Set disk variables and add files
        Select Case LCase(ScriptEntryType)
            Case "path:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                If ScriptEntryArray.Length > 0 Then
                    D64Name = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
                End If
                NewBundle = True
            Case "header:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                If ScriptEntryArray.Length > 0 Then
                    DiskHeader = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
                End If
```

```
                    NewBundle = True
                Case "id:"
                    If NewD = False Then
                        NewD = True
                        If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                        If ResetDiskVariables() = False Then GoTo NoDisk
                    End If
                    If ScriptEntryArray.Length > 0 Then
                        DiskID = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
                    End If
                    NewBundle = True
                Case "name:"
                    If NewD = False Then
                        NewD = True
                        If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                        If ResetDiskVariables() = False Then GoTo NoDisk
                    End If
                    If ScriptEntryArray.Length > 0 Then
                        DemoName = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
                    End If
                    NewBundle = True
                Case "start:"
                    If NewD = False Then
                        NewD = True
                        If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                        If ResetDiskVariables() = False Then GoTo NoDisk
                    End If
                    If ScriptEntryArray.Length > 0 Then
                        DemoStart = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
                    End If
                    NewBundle = True
                Case "dirart:"
                    If NewD = False Then
                        NewD = True
                        If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                        If ResetDiskVariables() = False Then GoTo NoDisk
                    End If
                    If (ScriptEntryArray.Length > 0) AndAlso (ScriptEntryArray(0) <> "") Then
                        If InStr(ScriptEntryArray(0), ":") = 0 Then
                            ScriptEntryArray(0) = ScriptPath + ScriptEntryArray(0)
                        End If
                        If IO.File.Exists(ScriptEntryArray(0)) Then
                            DirArtName = ScriptEntryArray(0)
                            'DirArt = IO.File.ReadAllText(DirArtName)
                        Else
                            MsgBox("The following DirArt file does not exist:" + vbNewLine + vbNewLine +
ScriptEntryArray(0), vbOKOnly + vbExclamation, "DirArt file not found")
                        End If
                    End If
                    NewBundle = True
                Case "zp:"
                    If NewD = False Then
                        NewD = True
                        If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                        If ResetDiskVariables() = False Then GoTo NoDisk
                    End If
                    If DiskCnt = 0 Then
                        If (ScriptEntryArray.Length > 0) AndAlso (ScriptEntryArray(0) IsNot Nothing) Then
                            LoaderZP = ScriptEntryArray(0)      'ZP usage can only be set from first disk
                        End If
                    End If
                    NewBundle = True
```

```
            Case "packer:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                If ScriptEntryArray.Length > 0 Then
                    If ScriptEntryArray(0) IsNot Nothing Then
                        Packer = If(LCase(ScriptEntryArray(0)) = "better", PackerTypes.Better, PackerTypes.Faster)
                    End If
                End If
                NewBundle = True
            Case "il0:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                Dim TmpIL As Integer = If(ScriptEntryArray.Length > 0, Convert.ToInt32(ScriptEntryArray(0), 10), 0)
                IL0 = If(TmpIL Mod 21 > 0, TmpIL Mod 21, DefaultIL0)
                NewBundle = True
            Case "il1:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                Dim TmpIL As Integer = If(ScriptEntryArray.Length > 0, Convert.ToInt32(ScriptEntryArray(0), 10), 0)
                IL1 = If(TmpIL Mod 19 > 0, TmpIL Mod 19, DefaultIL1)
                NewBundle = True
            Case "il2:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                Dim TmpIL As Integer = If(ScriptEntryArray.Length > 0, Convert.ToInt32(ScriptEntryArray(0), 10), 0)
                IL2 = If(TmpIL Mod 18 > 0, TmpIL Mod 18, DefaultIL2)
                NewBundle = True
            Case "il3:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                Dim TmpIL As Integer = If(ScriptEntryArray.Length > 0, Convert.ToInt32(ScriptEntryArray(0), 10), 0)
                IL3 = If(TmpIL Mod 17 > 0, TmpIL Mod 17, DefaultIL3)
                NewBundle = True
            Case "prodid:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                If ScriptEntryArray.Length > 0 Then
                    If IsNumeric("&H" + ScriptEntryArray(0)) Then
                        ProductID = Convert.ToInt32(ScriptEntryArray(0), 16)
                    Else
```

```
                        MsgBox("The Product ID must be a maximum 6-digit long hexadecimal number!" + vbNewLine + vbNewLine +
                               "Sparkle will generate a pseudorandom Product ID.", vbOKOnly + vbExclamation, "Product ID Error")
                    End If
                End If
                NewBundle = True
            Case "tracks:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                If ScriptEntryArray.Length > 0 Then
                    If IsNumeric(ScriptEntryArray(0)) Then
                        Dim TmpTracks As Integer = Convert.ToInt32(ScriptEntryArray(0), 10)
                        If TmpTracks = ExtTracksPerDisk Then
                            TracksPerDisk = ExtTracksPerDisk
                            SectorsPerDisk = ExtSectorsPerDisk
                        Else
                            TracksPerDisk = StdTracksPerDisk
                            SectorsPerDisk = StdSectorsPerDisk
                        End If
                        UpdateDiskSizeOnTheFly()
                    End If
                End If
                NewBundle = True
            Case "hsfile:"
                If NewD = False Then
                    NewD = True
                    If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
                    If ResetDiskVariables() = False Then GoTo NoDisk
                End If
                If ScriptEntryArray(0) <> "" Then
                    If AddHSFile() = False Then GoTo NoDisk
                End If
                NewBundle = True
            Case "script:"
                If InsertScript(ScriptEntryArray(0)) = False Then GoTo NoDisk
                NewBundle = True    'Files in the embedded script will ALWAYS be in a new bundle (i.e. scripts cannot be embedded in a bundle)!!!
            Case "file:"
                'Add files to bundle array, if new bundle=true, we will first sort, compress and add previous bundle to disk
                If AddFile() = False Then GoTo NoDisk
                NewD = False    'We have added at least one file to this disk, so next disk info entry will be a new disk
                NewBundle = False
            Case "mem:"
                If AddVirtualFile() = False Then GoTo NoDisk
                NewBundle = False
                'NewD = False           'IS THIS NEEDED???
            'Case "sort:"
            Case "align"
                If NewD = False Then
                    TmpSetNewBlock = True
                End If
            Case Else
                If NewBundle = True Then
                    If BundleDone() = False Then GoTo NoDisk
                    NewBundle = False
                End If
        End Select

        If SE < Script.Length Then GoTo FindNext

        'Last disk: sort, compress and add last bundle, then update bundle count, add loader & drive code, save disk, and we are done :)
        If FinishDisk(True, SaveIt) = False Then GoTo NoDisk

        'MsgBox(TotalBits.ToString)
        'MsgBox("Literals:" + vbTab + vbTab + vbTab + TotLits.ToString + vbNewLine +
        '"Short Matches:" + vbTab + vbTab + TotSM.ToString + vbNewLine +
        '"Near Mid Matches:" + vbTab + TotNMM.ToString + vbNewLine +
        '"Near Long Matches:" + vbTab + TotNLM.ToString + vbNewLine +
        '"Far Mid Matches:" + vbTab + vbTab + TotFMM.ToString + vbNewLine +
        '"Far Long Matches:" + vbTab + vbTab + TotFLM.ToString)

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")
NoDisk:
        BuildDemoFromScript = False
        If ErrCode = 0 Then ErrCode = -1

    End Function

    Public Function InsertScript(SubScriptPath As String) As Boolean
        If DoOnErr Then On Error GoTo Err

        InsertScript = True

        Dim SPath As String = SubScriptPath

        'Calculate full path
        If InStr(SubScriptPath, ":") = 0 Then SubScriptPath = ScriptPath + SubScriptPath

        If IO.File.Exists(SubScriptPath) = False Then
            MsgBox("The following script was not found and could not be processed:" + vbNewLine + vbNewLine + SubScriptPath, vbOKOnly + vbExclamation, "Script not found")
            InsertScript = False
            Exit Function
        End If

        'Find relative path of subscript
        For I As Integer = Len(SPath) - 1 To 0 Step -1
            If Right(SPath, 1) <> "\" Then
                SPath = Left(SPath, Len(SPath) - 1)
            Else
                Exit For
            End If
        Next

        'Find relative path of subscript - THIS DOESN'T WORK IF THERE IS NO "\" IN THE SUBSCRIPTS PATH
        'WOULD PROPABLY WORK WITH THE ADDITIONAL INSTR() CHECK BUT ANYWAY, LET'S USE THE OLD AND PROVEN CODE ABOVE...
        'If InStr(SPath, "\") <> 0 Then
        'For I As Integer = Len(SPath) To 1 Step -1
        'If Mid(SPath, I, 1) = "\" Then
        'SPath = Strings.Left(SPath, I)            'Path
        'Exit For
        'End If
        'Next
        'Else
        'SPath = ""
        'End If

        Dim Lines() As String = Split(IO.File.ReadAllText(SubScriptPath), vbLf)

        Dim S As String = ""
        For I As Integer = 0 To Lines.Count - 1
            Lines(I) = Lines(I).TrimEnd(Chr(13))    'Trim vbCR from end of lines if vbCrLf was used

            If InStr(Lines(I), vbTab) = 0 Then
                ScriptEntryType = Replace(Lines(I), "", "")
                ScriptEntry = ""
            Else
                ScriptEntryType = Replace(Strings.Left(Lines(I), InStr(Lines(I), vbTab) - 1), " ", "")
                ScriptEntry = Strings.Right(Lines(I), Len(Lines(I)) - InStr(Lines(I), vbTab)).TrimStart(vbTab)
            End If

            SplitEntry()

            'Skip Script Header
            If Lines(I) <> ScriptHeader Then
                If S <> "" Then
                    S += vbNewLine
                End If
                'Add relative path of subscript to relative path of subscript entries
                Select Case LCase(ScriptEntryType)
                    Case "file:"
                        If ScriptEntryArray(0) IsNot Nothing Then
                            If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath + ScriptEntryArray(0)
                        End If
                        Lines(I) = "File:" + vbTab + ScriptEntryArray(0)
                        For J As Integer = 1 To ScriptEntryArray.Length - 1
                            If ScriptEntryArray(J) IsNot Nothing Then
                                Lines(I) += vbTab + ScriptEntryArray(J)
                            End If
                        Next
                    Case "mem:"
                        If ScriptEntryArray(0) IsNot Nothing Then
                            If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath + ScriptEntryArray(0)
                        End If
                        Lines(I) = "Mem:" + vbTab + ScriptEntryArray(0)
                        For J As Integer = 1 To ScriptEntryArray.Length - 1
                            If ScriptEntryArray(J) IsNot Nothing Then
                                Lines(I) += vbTab + ScriptEntryArray(J)
                            End If
                        Next
                    Case "script:"
                        If ScriptEntryArray(0) IsNot Nothing Then
                            If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath + ScriptEntryArray(0)
                        End If
                        Lines(I) = "Script:" + vbTab + ScriptEntryArray(0)
                    Case "path:"
                        If ScriptEntryArray(0) IsNot Nothing Then
                            If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath + ScriptEntryArray(0)
                        End If
                        Lines(I) = "Path:" + vbTab + ScriptEntryArray(0)
                    Case "dirart:"
                        If ScriptEntryArray(0) IsNot Nothing Then
```

```
                            If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath + ScriptEntryArray(0)
                        End If
                        Lines(I) = "DirArt:" + vbTab + ScriptEntryArray(0)
                    Case "hsfile:"
                        If ScriptEntryArray(0) IsNot Nothing Then
                            If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath + ScriptEntryArray(0)
                        End If
                        Lines(I) = "HSFile:" + vbTab + ScriptEntryArray(0)
                        For J As Integer = 1 To ScriptEntryArray.Length - 1
                            If ScriptEntryArray(J) IsNot Nothing Then
                                Lines(I) += vbTab + ScriptEntryArray(J)
                            End If
                        Next
                End Select
                'If Strings.Right(Lines(I), 1) <> vbLf Then
                'Lines(I) += vbLf
                'End If


                'If Left(LCase(Lines(I)), 5) = "file:" Then
                'If (InStr(Right(Lines(I), Len(Lines(I)) - 5), ":") = 0) And (InStr(Right(Lines(I), Len(Lines(I)) - 5), SPath) = 0) Then
                'Lines(I) = "File:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) - 5).TrimStart(vbTab)    'Trim any extra leading TABs
                'End If
                'ElseIf Left(LCase(Lines(I)), 7) = "script:" Then
                'If (InStr(Right(Lines(I), Len(Lines(I)) - 7), ":") = 0) And (InStr(Right(Lines(I), Len(Lines(I)) - 7), SPath) = 0) Then
                'Lines(I) = "Script:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) - 7).TrimStart(vbTab)
                'End If
                'ElseIf Left(LCase(Lines(I)), 5) = "list:" Then
                'If (InStr(Right(Lines(I), Len(Lines(I)) - 5), ":") = 0) And (InStr(Right(Lines(I), Len(Lines(I)) - 5), SPath) = 0) Then
                'Lines(I) = "Script:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) - 5).TrimStart(vbTab)
                'End If
                'ElseIf Left(LCase(Lines(I)), 5) = "path:" Then
                'If (InStr(Right(Lines(I), Len(Lines(I)) - 5), ":") = 0) And (InStr(Right(Lines(I), Len(Lines(I)) - 5), SPath) = 0) Then
                'Lines(I) = "Path:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) - 5).TrimStart(vbTab)
                'End If
                'ElseIf Left(LCase(Lines(I)), 7) = "dirart:" Then
                'If (InStr(Right(Lines(I), Len(Lines(I)) - 7), ":") = 0) And (InStr(Right(Lines(I), Len(Lines(I)) - 7), SPath) = 0) Then
                'Lines(I) = "DirArt:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) - 7).TrimStart(vbTab)
                'End If
                'ElseIf Left(LCase(Lines(I)), 7) = "hsfile:" Then
                'If (InStr(Right(Lines(I), Len(Lines(I)) - 7), ":") = 0) And (InStr(Right(Lines(I), Len(Lines(I)) - 7), SPath) = 0) Then
                'Lines(I) = "HSFile:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) - 7).TrimStart(vbTab)
                'End If
                'End If
                S += Lines(I)
            End If
        Next

        Script = Replace(Script, ScriptLine, S)

        SS = LastSS
        SE = LastSE

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

        InsertScript = False

    End Function

    Private Function AddHeaderAndID() As Boolean
        If DoOnErr Then On Error GoTo Err

        AddHeaderAndID = True

        Dim B As Byte

        CP = Track(18)

        For Cnt As Integer = &H90 To &HAA
            Disk(CP + Cnt) = &HA0
        Next

        If Len(DiskHeader) > 16 Then
            DiskHeader = Left(DiskHeader, 16)
        End If

        If DiskHeader = "" Then
            For Cnt As Integer = 1 To 16
                Disk(CP + &H8F + Cnt) = 32
            Next
        Else
            For Cnt As Integer = 1 To Len(DiskHeader)
                B = Ascii2Petscii(Asc(Mid(DiskHeader, Cnt, 1)))
                'If B > &H5F Then B -= &H20
                Disk(CP + &H8F + Cnt) = B
            Next
        End If

        If Len(DiskID) > 5 Then
            DiskID = Left(DiskID, 5)
        End If

        If DiskID = "" Then
            For Cnt As Integer = 1 To 5                           'Overwrites Disk ID and DOS type (5
characters max.)
                Disk(CP + &HA1 + Cnt) = 32
            Next
        Else
            For Cnt As Integer = 1 To Len(DiskID)                  'Overwrites Disk ID and DOS type (5
characters max.)
                B = Ascii2Petscii(Asc(Mid(DiskID, Cnt, 1)))
                'If B > &H5F Then B -= &H20
                Disk(CP + &HA1 + Cnt) = B
            Next
        End If


        Exit Function
```

```
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")

        AddHeaderAndID = False

    End Function

    Private Function FinishDisk(LastDisk As Boolean, Optional SaveIt As Boolean = True) As Boolean
        If DoOnErr Then On Error GoTo Err

        FinishDisk = True

        If (BundleCnt = 0) And (FileCnt = -1) Then
            MsgBox("This disk does not contain any files!", vbOKOnly + vbExclamation, "Unable to build disk")
            GoTo NoDisk
        End If
        If BundleDone() = False Then GoTo NoDisk
        If CompressBundle() = False Then GoTo NoDisk
        If CloseBundle(0, True) = False Then GoTo NoDisk
        If CloseBuffer() = False Then GoTo NoDisk

        If MaxBundleNoExceeded Then
            MsgBox("The number of file bundles is greater than 128 on this disk!" + vbNewLine + vbNewLine +
               "You can only access bundles 0-127 by bundle index." + vbNewLine + "The rest can only be loaded using the LoadNext function.", vbOKOnly + vbInformation, "More than 128 bundles on disk")
        End If

        'Now add compressed parts to disk
        If AddCompressedBundlesToDisk() = False Then GoTo NoDisk
        If InjectLoader(-1, 18, 7, 1) = False Then GoTo NoDisk
        If InjectDriveCode(DiskCnt, LoaderBundles, If(LastDisk = False, DiskCnt + 1, &H80)) = False Then GoTo NoDisk

        AddHeaderAndID()
        AddDemoNameToDisk(-1, 18, 7)
        AddDirArt()

        BytesSaved += Int(BitsSaved / 8)
        BitsSaved = BitsSaved Mod 8

        UpdateBlocksFree()

        If SaveIt = True Then
            If SaveDisk() = False Then GoTo NoDisk
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")
NoDisk:
        FinishDisk = False

    End Function

    Private Function SaveDisk() As Boolean
        'CANNOT HAVE On Error FUNCTION DUE TO TRY/CATCH

        SaveDisk = True

        If D64Name = "" Then D64Name = "Demo Disk " + (DiskCnt + 1).ToString + ".d64"

        If InStr(D64Name, ":") = 0 Then
            D64Name = ScriptPath + D64Name
        End If

        Dim SaveCtr As Integer = 20

TryAgain:
        ErrCode = 0
        Try
            If CmdLine = True Then
                'We are in command line, just save the disk
                IO.File.WriteAllBytes(D64Name, Disk)
            Else
                'We are in app mode, show dialog
                Dim SaveDLG As New SaveFileDialog With {
                .Filter = "D64 Files (*.d64)|*.d64",
                .Title = "Save D64 File As...",
                .FileName = D64Name,
                .RestoreDirectory = True
            }

                Dim R As DialogResult = SaveDLG.ShowDialog(FrmMain)

                If R = DialogResult.OK Then
                    D64Name = SaveDLG.FileName
                    If Right(D64Name, 4) <> ".d64" Then
                        D64Name += ".d64"
                    End If
                    IO.File.WriteAllBytes(D64Name, Disk)
                    FileChanged = False
                Else
                    FileChanged = True
                End If

            End If
        Catch ex As Exception
            If CmdLine = True Then
                If SaveCtr > 0 Then
                    Threading.Thread.Sleep(20)  'If file could not be saved, wait 20 msec and try
again 20 times before showing error message
                    SaveCtr -= 1
                    GoTo TryAgain
                End If
            End If
            ErrCode = Err.Number    'Save error code here
            If MsgBox(ex.Message + vbNewLine + "Error code:  " + Err.Number.ToString + vbNewLine +
vbNewLine + "Do you want to try again?", vbYesNo + vbExclamation,
Reflection.MethodBase.GetCurrentMethod.Name + " Error") = vbYes Then
                Err.Clear()
                SaveCtr = 20
                GoTo TryAgain
            Else
                SaveDisk = False
                FileChanged = True
            End If
        End Try

    End Function

    Public Function CompressBundle(Optional FromEditor = False) As Boolean
        If DoOnErr Then On Error GoTo Err

        CompressBundleFromEditor = FromEditor

        CompressBundle = True

        Dim PreBCnt As Integer = BufferCnt

        If Prgs.Count = 0 Then Exit Function        'GoTo NoComp DOES NOT WORK!!!

        'DO NOT RESET ByteSt AND BUFFER VARIABLES HERE!!!

        If (BufferCnt = 0) And (BytePtr = 255) Then
            NewBlock = SetNewBlock          'SetNewBlock is true at closing the previous bundle, so first it just sets NewBlock2
            SetNewBlock = False             'And NewBlock will fire at the desired bundle
        Else
            If FromEditor = False Then      'Don't finish previous bundle here if we are calculating bundle size from Editor

                '-----------------------------------------------------------------------------------
                '"SPRITE BUG"
                'Compression bug involving the transitional block - FIXED
                'Fix: include the I/O status of the first file of this bundle in the calculation for
                'finishing the previous bundle
                '-----------------------------------------------------------------------------------

                'Before finishing the previous bundle, calculate I/O status of the LAST BYTE of the first file of this bundle
                '(Files already sorted)
                Dim ThisBundleIO As Integer = If(FileIOA.Count > 0, CheckNextIO(FileAddrA(0), FileLenA(0), FileIOA(0)), 0)
                If CloseBundle(ThisBundleIO, False) = False Then GoTo NoComp
            End If
        End If

        '-------------------------------------------------------
        'SAVE CURRENT BIT POINTER AND BUFFER COUNT FOR DIRECTORY
        '-------------------------------------------------------

        If FromEditor = False Then
            'Only if we are NOT in the Editor
            If BundleNo < 128 Then
                DirBlocks((BundleNo * 4) + 3) = BitPtr
                DirPtr(BundleNo) = BufferCnt
                BundleNo += 1
            Else
                MaxBundleNoExceeded = True
            End If
        End If

        '-------------------------------------------------------

        NewBundle = True
        LastFileOfBundle = False

        PartialFileIndex = -1

        For I As Integer = 0 To Prgs.Count - 1
            'Mark the last file in a bundle for better compression
            If I = Prgs.Count - 1 Then LastFileOfBundle = True
            'The only two parameters that are needed are FA and FUIO... FileLenA(i) is not used

            If PartialFileIndex = -1 Then PartialFileOffset = Prgs(I).ToArray.Length - 1

            PackFile(Prgs(I).ToArray, I, FileAddrA(I), FileIOA(I))
            If I < Prgs.Count - 1 Then
                'WE NEED TO USE THE NEXT FILE'S ADDRESS, LENGTH AND I/O STATUS HERE
                'FOR I/O BYTE CALCULATION FOR THE NEXT PART - BUG reported by Raistlin/G*P
                PrgAdd = Convert.ToInt32(FileAddrA(I + 1), 16)
                PrgLen = Prgs(I + 1).Length ' Convert.ToInt32(FileLenA(I + 1), 16)
                FileUnderIO = FileIOA(I + 1)
                CloseFile()
            End If
        Next

        LastBlockCnt = BlockCnt

        If LastBlockCnt > 255 Then
            'Parts cannot be larger than 255 blocks compressed
            'There is some confusion here how PartCnt is used in the Editor and during Disk
building...
            MsgBox("Bundle " + If(CompressBundleFromEditor = True, BundleCnt + 1, BundleCnt).ToString
+ " would need " + LastBlockCnt.ToString + " blocks on the disk." + vbNewLine + vbNewLine + "Bundles
cannot be larger than 255 blocks compressed!", vbOKOnly + vbCritical, "Bundle exceeds 255-block
limit!")
            If CompressBundleFromEditor = False Then GoTo NoComp
        End If

        'IF THE WHOLE Bundle IS LESS THAN 1 BLOCK, THEN "IT DOES NOT COUNT", Bundle Counter WILL NOT
BE INCREASED
        If PreBCnt = BufferCnt Then
            BundleCnt -= 1
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
NoComp:
        CompressBundle = False

    End Function

    Private Function AddFile() As Boolean
        If DoOnErr Then On Error GoTo Err

        AddFile = True

        If NewBundle = True Then
            If BundleDone() = False Then GoTo NoDisk
        End If

        'Then add file to bundle
        If AddFileToBundle() = False Then GoTo NoDisk

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
NoDisk:
        AddFile = False

    End Function
    Private Function BundleDone() As Boolean
        If DoOnErr Then On Error GoTo Err

        BundleDone = True

        'First finish last bundle, if it exists
        If tmpPrgs.Count > 0 Then

            CurrentBundle += 1

            'Sort files in bundle
            If SortBundle() = False Then GoTo NoDisk
            '-------------------------------------------
            'Then compress files and add them to bundle
            If CompressBundle() = False Then GoTo NoDisk     'THIS WILL RESET NewPart TO FALSE

            Prgs = tmpPrgs.ToList
            FileNameA = tmpFileNameA
            FileAddrA = tmpFileAddrA
            FileOffsA = tmpFileOffsA
            FileLenA = tmpFileLenA
            FileIOA = tmpFileIOA
            SetNewBlock = TmpSetNewBlock
            TmpSetNewBlock = False

            VFiles = tmpVFiles.ToList
            VFileNameA = tmpVFileNameA
            VFileAddrA = tmpVFileAddrA
            VFileOffsA = tmpVFileOffsA
            VFileLenA = tmpVFileLenA
            VFileIOA = tmpVFileIOA

            '-------------------------------------------
            'Then reset bundle variables (file arrays, prg array, block cnt), increase bundle counter
            ResetBundleVariables()
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
NoDisk:
        BundleDone = False

    End Function

    Public Function CheckNextIO(sAddress As String, sLength As String, NextFileUnderIO As Boolean) As
Integer
        If DoOnErr Then On Error GoTo Err

        Dim pAddress As Integer = Convert.ToInt32(sAddress, 16) + Convert.ToInt32(sLength, 16)

        If pAddress < 256 Then       'Are we loading to the Zero Page? If yes, we need to signal it by
adding IO Flag
            CheckNextIO = 1
        Else
            CheckNextIO = If((pAddress >= &HD000) And (pAddress <= &HDFFF) And (NextFileUnderIO =
True), 1, 0)
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")

    End Function

    Public Function SortBundle() As Boolean
        If DoOnErr Then On Error GoTo Err

        SortBundle = True

        If tmpPrgs.Count = 0 Then Exit Function
        If tmpPrgs.Count = 1 Then GoTo SortDone

        Dim Change As Boolean
        Dim FSO, FEO, FSI, FEI As Integer   'File Start and File End Outer loop/Inner loop
        Dim PO(), PI() As Byte
        Dim S As String
        Dim bIO As Boolean

        '--------------------------------------------------------------------------------
        'Check files for overlap

        For O As Integer = 0 To tmpPrgs.Count - 2
            FSO = Convert.ToInt32(tmpFileAddrA(O), 16)              'Outer loop File Start
            FEO = FSO + Convert.ToInt32(tmpFileLenA(O), 16) - 1     'Outer loop File End
            For I As Integer = O + 1 To tmpPrgs.Count - 1
                FSI = Convert.ToInt32(tmpFileAddrA(I), 16)          'Inner loop File Start
                FEI = FSI + Convert.ToInt32(tmpFileLenA(I), 16) - 1 'Inner loop File End
                '--|------+------|----OR----|------+------|----OR----|------+------|----OR-----|------+------|--
                '  FSO    FSI    FEO        FSO    FEI    FEO        FSI    FSO    FEI        FSI    FEO    FEI
                If ((FSI >= FSO) And (FSI <= FEO)) Or ((FEI >= FSO) And (FEI <= FEO)) Or ((FSO >= FSI) And (FSO <= FEI)) Or ((FEO >= FSI) And (FEO <= FEI)) Then
                    Dim OLS As Integer = If(FSO >= FSI, FSO, FSI)  'Overlap Start address
                    Dim OLE As Integer = If(FEO <= FEI, FEO, FEI)  'Overlap End address

                    If (OLS >= &HD000) And (OLE <= &HDFFF) And (tmpFileIOA(O) <> tmpFileIOA(I)) Then
                        'Overlap is IO memory only and different IO status - NO OVERLAP
                    Else
                        MsgBox("The following two files overlap in Bundle " + (BundleCnt - 1).ToString + ":" _
                           + vbNewLine + vbNewLine + tmpFileNameA(I) + " ($" + Hex(FSI) + " - $" + Hex(FEI) + ")" + vbNewLine + vbNewLine _
                           + tmpFileNameA(O) + " ($" + Hex(FSO) + " - $" + Hex(FEO) + ")", vbOKOnly + vbExclamation)
                    End If
                End If
            Next
        Next

        '--------------------------------------------------------------------------------
        'Append adjacent files
Restart:
        Change = False

        For O As Integer = 0 To tmpPrgs.Count - 2
            FSO = Convert.ToInt32(tmpFileAddrA(O), 16)
            FEO = Convert.ToInt32(tmpFileLenA(O), 16)
            For I As Integer = O + 1 To tmpPrgs.Count - 1
```

```
                FSI = Convert.ToInt32(tmpFileAddrA(I), 16)
                FEI = Convert.ToInt32(tmpFileLenA(I), 16)

                If FSO + FEO = FSI Then
                    'Inner file follows outer file immediately
                    If (FSI <= &HD000) Or (FSI > &HDFFF) Then
                        'Append files as they meet outside IO memory
Append:                 PO = tmpPrgs(O)
                        PI = tmpPrgs(I)
                        ReDim Preserve PO(FEO + FEI - 1)

                        For J As Integer = 0 To FEI - 1
                            PO(FEO + J) = PI(J)
                        Next

                        tmpPrgs(O) = PO

                        Change = True
                    Else
                        If tmpFileIOA(O) = tmpFileIOA(I) Then
                            'Files meet inside IO memory, append only if their IO status is the same
                            GoTo Append
                        End If
                    End If
                ElseIf FSI + FEI = FSO Then
                    'Outer file follows inner file immediately
                    If (FSO <= &HD000) Or (FSO > &HDFFF) Then
                        'Prepend files as they meet outside IO memory
Prepend:                PO = tmpPrgs(O)
                        PI = tmpPrgs(I)
                        ReDim Preserve PI(FEI + FEO - 1)

                        For J As Integer = 0 To FEO - 1
                            PI(FEI + J) = PO(J)
                        Next

                        tmpPrgs(O) = PI

                        tmpFileAddrA(O) = tmpFileAddrA(I)

                        Change = True
                    Else
                        If tmpFileIOA(O) = tmpFileIOA(I) Then
                            'Files meet inside IO memory, prepend only if their IO status is the same
                            GoTo Prepend
                        End If
                    End If
                End If

                If Change = True Then
                    'Update merged file's IO status
                    tmpFileIOA(O) = tmpFileIOA(O) Or tmpFileIOA(I)   'BUG FIX - REPORTED BY
RAISTLIN/G*P
                    'New file's length is the length of the two merged files
                    FEO += FEI

                    tmpFileLenA(O) = ConvertIntToHex(FEO, 4)
                    'Remove File(I) and all its parameters
                    For J As Integer = I To tmpPrgs.Count - 2
                        tmpFileNameA(J) = tmpFileNameA(J + 1)
                        tmpFileAddrA(J) = tmpFileAddrA(J + 1)
                        tmpFileOffsA(J) = tmpFileOffsA(J + 1)     'this may not be needed later
                        tmpFileLenA(J) = tmpFileLenA(J + 1)
                        tmpFileIOA(J) = tmpFileIOA(J + 1)
                    Next
                    'One less file left
                    FileCnt -= 1
                    ReDim Preserve tmpFileNameA(tmpPrgs.Count - 2), tmpFileAddrA(tmpPrgs.Count - 2), tmpFileOffsA(tmpPrgs.Count - 2), tmpFileLenA(tmpPrgs.Count - 2)
                    ReDim Preserve tmpFileIOA(tmpPrgs.Count - 2)
                    tmpPrgs.Remove(tmpPrgs(I))
                    GoTo Restart
                End If
            Next
        Next

        '-------------------------------------------------------------------------------
        'Sort files by length (short files first, thus, last block will more likely contain 1 file only = faster depacking)
ReSort:
        Change = False
        For I As Integer = 0 To tmpPrgs.Count - 2
            'Sort except if file length < 4, to allow for ZP relocation script hack
            If Convert.ToInt32(tmpFileAddrA(I), 16) < Convert.ToInt32(tmpFileAddrA(I + 1), 16) Then
                PI = tmpPrgs(I)
                tmpPrgs(I) = tmpPrgs(I + 1)
                tmpPrgs(I + 1) = PI

                S = tmpFileNameA(I)
                tmpFileNameA(I) = tmpFileNameA(I + 1)
                tmpFileNameA(I + 1) = S

                S = tmpFileAddrA(I)
                tmpFileAddrA(I) = tmpFileAddrA(I + 1)
                tmpFileAddrA(I + 1) = S

                S = tmpFileOffsA(I)
                tmpFileOffsA(I) = tmpFileOffsA(I + 1)
                tmpFileOffsA(I + 1) = S

                S = tmpFileLenA(I)
                tmpFileLenA(I) = tmpFileLenA(I + 1)
                tmpFileLenA(I + 1) = S

                bIO = tmpFileIOA(I)
                tmpFileIOA(I) = tmpFileIOA(I + 1)
                tmpFileIOA(I + 1) = bIO
                Change = True
            End If
        Next
        If Change = True Then GoTo ReSort

SortDone:
        'Once Bundle is sorted, calculate the I/O status of the last byte of the first file and the number of bits that will be needed
        'to finish the last block of the previous bundle (when the I/O status of the just sorted bundle needs to be known)
        'This is used in CloseBuffer

        'Bytes needed: (1)LongMatch Tag, (2)NextBundle Tag, (3)AdLo, (4)AdHi, (5)First Lit, (6)1 Bit Stream Byte (for 1 Lit Bit), (7)+/- I/O
        '+/- 1 Match Bit (if the last sequence of the last bundle is a match sequence, no Match Bit after a Literal sequence)
        'Match Bit will be determened by MLen in SequenceFits() function, NOT ADDED TO BitsNeededForNextBundle here!!!
        'We may be overcalculating here but that is safer than undercalculating which would result in buggy decompression
        'If the last block is not the actual last block of the bundle...
        'With overcalculation, worst case scenario is a little bit worse compression ratio of the last block
        BitsNeededForNextBundle = (6 + CheckNextIO(tmpFileAddrA(0), tmpFileLenA(0), tmpFileIOA(0))) * 8
        ' +/- 1 Match Bit which will be added later in CloseBuffer if needed

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")
NoSort:
        SortBundle = False

    End Function

    Public Function AddHSFile() As Boolean
        If DoOnErr Then On Error GoTo Err

        AddHSFile = True

        Dim FN As String = ScriptEntryArray(0)
        Dim FA As String = ""
        Dim FO As String = ""
        Dim FL As String = ""
        Dim FAN As Integer = 0
        Dim FON As Integer = 0
        Dim FLN As Integer = 0

        Dim NumParams As Integer = 1

        Dim P() As Byte

        If InStr(FN, ":") = 0 Then          'relative file path
            FN = ScriptPath + FN            'look for file in script's folder
        End If

        'Correct file parameter length to 4-8 characters
        For I As Integer = 1 To ScriptEntryArray.Count - 1

            If ParameterIsNumeric(I) Then
                NumParams += 1
            Else
                Exit For
            End If

            CorrectParameterStringLength(I)

            ''Remove HEX prefix
            'If Left(ScriptEntryArray(I), 1) = "$" Then
            'ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 1)
            'End If
            'Select Case LCase(Left(ScriptEntryArray(I), 2))
            'Case "&h", "0x"
            'ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 2)
            'End Select

            'If Left(ScriptEntryArray(I), 1) = "." Then
            'ScriptEntryArray(I) = ScriptEntryArray(I).TrimStart(".")
            'If IsNumeric(ScriptEntryArray(I)) Then
            'Dim ScriptEntryInt As Integer = Convert.ToInt32(ScriptEntryArray(I))
            'ScriptEntryArray(I) = Hex(ScriptEntryInt)
            'Else
            'Exit For
            'End If
            'End If

            'If IsNumeric("&H" + ScriptEntryArray(I)) Then
            ''If IsHexString(ScriptEntryArray(I)) Then
            'NumParams = I + 1
            'Else
            'Exit For
            'End If

            ''Remove unwanted spaces
            'Replace(ScriptEntryArray(I), " ", "")

            'Select Case I
            'Case 2      'File Offset max. $ffff ffff (dword)
            'If Len(ScriptEntryArray(I)) < 8 Then
            'ScriptEntryArray(I) = Left("00000000", 8 - Len(ScriptEntryArray(I))) + ScriptEntryArray(I)
            'ElseIf (I = 2) And (len(ScriptEntryArray(I)) > 8) Then
            'ScriptEntryArray(I) = Right(ScriptEntryArray(I), 8)
            'End If
            'Case Else   'File Address, File Length max. $ffff
            'If Len(ScriptEntryArray(I)) < 4 Then
            'ScriptEntryArray(I) = Left("0000", 4 - Len(ScriptEntryArray(I))) + ScriptEntryArray(I)
            'ElseIf Len(ScriptEntryArray(I)) > 4 Then
            'ScriptEntryArray(I) = Right(ScriptEntryArray(I), 4)
            'End If
            'End Select
        Next

        'Get file variables from script, or get default values if there were none in the script entry
        If IO.File.Exists(Replace(FN, "*", "")) = True Then
            P = IO.File.ReadAllBytes(Replace(FN, "*", ""))

            Select Case NumParams   'ScriptEntryArray.Count
                Case 1  'No parameters in script
                    If InStr(LCase(Replace(FN, "*", "")), ".sid") <> 0 Then   'SID file - read parameters from file
                        FA = ConvertIntToHex(P(P(7)) + (P(P(7) + 1) * 256), 4)
                        FO = ConvertIntToHex(P(7) + 2, 8)
                        FL = ConvertIntToHex((P.Length - P(7) - 2), 4)
                    Else                                                    'Any other files
                        If P.Length > 2 Then                                'We have at least 3 bytes in the file
                            FA = ConvertIntToHex(P(0) + (P(1) * 256), 4)    'First 2 bytes define load address
                            FO = "00000002"                                 'Offset=2, Length=prg length-2
                            FL = ConvertIntToHex(P.Length - 2, 4)
                        Else                                                'Short file without paramters -> STOP
                            MsgBox("File parameters are needed for the following file:" + vbNewLine + vbNewLine + FN, vbCritical + vbOKOnly, "Missing file parameters")
                            GoTo NoDisk
                        End If
                    End If
                Case 2  'One parameter in script
                    FA = ScriptEntryArray(1)                                'Load address from script
                    FO = "00000000"                                         'Offset will be 0, length=prg length
                    FL = ConvertIntToHex(P.Length, 4)
                Case 3  'Two parameters in script
                    FA = ScriptEntryArray(1)                                'Load address from script
                    FO = ScriptEntryArray(2)                                'Offset from script
                    FON = Convert.ToInt32(FO, 16)                           'Make sure offset is valid
                    If FON > P.Length - 1 Then
                        FON = P.Length - 1                                  'If offset>prg length-1 then correct it
                        FO = ConvertIntToHex(FON, 8)
                    End If                                                  'Length=prg length-offset
                    FL = ConvertIntToHex(P.Length - FON, 4)
                Case 4  'Three parameters in script
                    FA = ScriptEntryArray(1)
                    FO = ScriptEntryArray(2)
                    FON = Convert.ToInt32(FO, 16)                           'Make sure offset is valid
                    If FON > P.Length - 1 Then
                        FON = P.Length - 1                                  'If offset>prg length-1 then correct it
                        FO = ConvertIntToHex(FON, 8)
                    End If                                                  'Length=prg length-offset
                    FL = ScriptEntryArray(3)
            End Select

            FAN = Convert.ToInt32(FA, 16)
            FON = Convert.ToInt32(FO, 16)
            FLN = Convert.ToInt32(FL, 16)

            'Make sure file length is not longer than actual file (should not happen)
            'If FON + FLN > P.Length Then
            'FLN = P.Length - FON
            'End If

            'Make sure file address+length<=&H10000
            If FAN + FLN > &H10000 Then
                FLN = (&H10000 - FAN) And &HF00
                If FLN < &H100 Then
                    MsgBox("The Hi-Score File's size must be at least $100 bytes!", vbOKOnly + vbExclamation, "Hi-Score File Error")
                    GoTo NoDisk
                End If
            End If

            'Round UP to nearest $100, at least $100 but not more than $0f00 bytes
            FLN = If((FLN Mod &H100 <> 0) Or (FLN = 0), FLN + &H100, FLN) And &HF00

            FL = ConvertIntToHex(FLN, 4)

            'Trim file to the specified chunk (FLN number of bytes starting at FON, to Address of FAN)
            Dim PL As List(Of Byte) = P.ToList      'Copy array to list
            P = PL.Skip(FON).Take(FLN).ToArray      'Trim file to specified segment (FLN number of bytes starting at FON)

            If P.Length < FLN Then
                ReDim Preserve P(FLN - 1)           'Round length up to nearest $100
            End If

            HSFile = P
            HSFileName = FN
            HSAddress = FAN
            HSOffset = FON
            HSLength = FLN
            bSaverPlugin = True

        Else
            'Add code here to create blank HSFile here if all 3 parameters are present
            If ScriptEntryArray.Count = 4 Then
                FA = ScriptEntryArray(1)
                FO = ScriptEntryArray(2)
                FL = ScriptEntryArray(3)

                FAN = Convert.ToInt32(FA, 16)
                FON = Convert.ToInt32(FO, 16)
                FLN = Convert.ToInt32(FL, 16)

                'Make sure file address+length<=&H10000
                If FAN + FLN > &H10000 Then
                    FLN = (&H10000 - FAN) And &HF00
                    If FLN < &H100 Then
                        MsgBox("The Hi-Score File's size must be at least $100 bytes!", vbOKOnly +
vbExclamation, "Hi-Score File Error")
                        GoTo NoDisk
                    End If
                End If

                'Round UP to nearest $100, at least $100 but not more than $0f00 bytes
                FLN = If((FLN Mod &H100 <> 0) Or (FLN = 0), FLN + &H100, FLN) And &HF00

                FL = ConvertIntToHex(FLN, 4)

                ReDim P(FLN - 1)    'Create blank HSFile based on file parameters

                HSFile = P
                HSFileName = FN
                HSAddress = FAN
                HSOffset = FON
                HSLength = FLN

                bSaverPlugin = True

            Else
                MsgBox("The following Hi-Score File does not exist:" + vbNewLine + vbNewLine + FN,
vbOKOnly + vbCritical, "Hi-Score File not found")
                GoTo NoDisk
            End If

        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
NoDisk:
        AddHSFile = False

    End Function

    Private Function ParameterIsNumeric(I As Integer) As Boolean
        If DoOnErr Then On Error GoTo Err

        'Remove unwanted spaces
        ScriptEntryArray(I) = Replace(ScriptEntryArray(I), " ", "")

        'Remove HEX prefix
        If Left(ScriptEntryArray(I), 1) = "$" Then
            ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 1)
        End If

        Select Case LCase(Left(ScriptEntryArray(I), 2))
            Case "&h", "0x"
                ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 2)
        End Select

        'If decimal -> convert it to hex
        If Left(ScriptEntryArray(I), 1) = "." Then
            ScriptEntryArray(I) = ScriptEntryArray(I).TrimStart(".")
            If IsNumeric(ScriptEntryArray(I)) Then
                Dim ScriptEntryInt As Integer = Convert.ToInt32(ScriptEntryArray(I))
                ScriptEntryArray(I) = Hex(ScriptEntryInt)
            Else
                ParameterIsNumeric = False
                Exit Function
            End If
        End If

        ParameterIsNumeric = IsNumeric("&H" + ScriptEntryArray(I))

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")


    End Function

    Private Sub CorrectParameterStringLength(I As Integer)
        If DoOnErr Then On Error GoTo Err

        Select Case I
            Case 2      'File Offset max. $ffff ffff (dword)
                If Len(ScriptEntryArray(I)) < 8 Then
                    ScriptEntryArray(I) = Left("00000000", 8 - Len(ScriptEntryArray(I))) +
ScriptEntryArray(I)
                ElseIf (I = 2) And (Len(ScriptEntryArray(I)) > 8) Then
                    ScriptEntryArray(I) = Right(ScriptEntryArray(I), 8)
                End If
            Case Else   'File Address, File Length max. $ffff
                If Len(ScriptEntryArray(I)) < 4 Then
                    ScriptEntryArray(I) = Left("0000", 4 - Len(ScriptEntryArray(I))) +
ScriptEntryArray(I)
                ElseIf Len(ScriptEntryArray(I)) > 4 Then
                    ScriptEntryArray(I) = Right(ScriptEntryArray(I), 4)
                End If
        End Select

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Function AddVirtualFile() As Boolean
```

```
        If DoOnErr Then On Error GoTo Err

        AddVirtualFile = True

        If NewBundle = True Then
            NewBundle = False
            If BundleDone() = False Then GoTo NoDisk
        End If

        Dim FN As String = ScriptEntryArray(0)
        Dim FA As String = ""
        Dim FO As String = ""
        Dim FL As String = ""
        Dim FAN As Integer
        Dim FON As Integer
        Dim FLN As Integer
        Dim FUIO As Boolean = False

        Dim NumParams As Integer = 1

        Dim P() As Byte

        If Right(FN, 1) = "*" Then
            FN = Replace(FN, "*", "")
            FUIO = True
        End If

        If InStr(FN, ":") = 0 Then          'relative file path
            FN = ScriptPath + FN            'look for file in script's folder
        End If

        'Correct file parameter lengths to 4-8 characters
        For I As Integer = 1 To ScriptEntryArray.Count - 1

            If ParameterIsNumeric(I) Then
                NumParams += 1
            Else
                Exit For
            End If

            CorrectParameterStringLength(I)

        Next

        'Get file variables from script, or get default values if there were none in the script entry
        If IO.File.Exists(FN) = True Then
            P = IO.File.ReadAllBytes(FN)

            Select Case NumParams   'ScriptEntryArray.Count
                Case 1  'No parameters in script
                    If InStr(LCase(FN), ".sid") <> 0 Then   'SID file - read parameters from file
                        FA = ConvertIntToHex(P(P(7)) + (P(P(7) + 1) * 256), 4)
                        FO = ConvertIntToHex(P(7) + 2, 8)
                        FL = ConvertIntToHex((P.Length - P(7) - 2), 4)
                    Else                                                    'Any other files
                        If P.Length > 2 Then                                'We have at least 3 bytes in the file
                            FA = ConvertIntToHex(P(0) + (P(1) * 256), 4)        'First 2 bytes define load address
                            FO = "00000002"                                     'Offset=2, Length=prg length-2
                            FL = ConvertIntToHex(P.Length - 2, 4)
                        Else                                                 'Short file without paramters -> STOP
                            MsgBox("File parameters are needed for the following file:" + vbNewLine + vbNewLine + FN, vbCritical + vbOKOnly, "Missing file parameters")
                            GoTo NoDisk
                        End If
                    End If
                Case 2  'One parameter in script
                    FA = ScriptEntryArray(1)                                 'Load address from script
                    FO = "00000000"                                          'Offset will be 0, length=prg length
                    FL = ConvertIntToHex(P.Length, 4)
                Case 3  'Two parameters in script
                    FA = ScriptEntryArray(1)                                 'Load address from script
                    FO = ScriptEntryArray(2)                                 'Offset from script
                    FON = Convert.ToInt32(FO, 16)                            'Make sure offset is valid
                    If FON > P.Length - 1 Then
                        FON = P.Length - 1                                   'If offset>prg length-1 then correct it
                        FO = ConvertIntToHex(FON, 8)
                    End If                                                   'Length=prg length- offset
                    FL = ConvertIntToHex(P.Length - FON, 4)
                Case 4  'Three parameters in script
                    FA = ScriptEntryArray(1)
                    FO = ScriptEntryArray(2)
                    FON = Convert.ToInt32(FO, 16)                            'Make sure offset is valid
                    If FON > P.Length - 1 Then
                        MsgBox("Invalid offset detected in the following entry:" + vbNewLine + vbNewLine +
                               ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid offset")
                        GoTo NoDisk
                        'FON = P.Length - 1                                   'If offset>prg length-1 then correct it
                        'FO = ConvertIntToHex(FON, 8)
                    End If                                                   'Length=prg length- offset
                    FL = ScriptEntryArray(3)
            End Select

            FAN = Convert.ToInt32(FA, 16)
            FON = Convert.ToInt32(FO, 16)
            FLN = Convert.ToInt32(FL, 16)

            'Make sure file length is not longer than actual file (should not happen)
            If FON + FLN > P.Length Then
                MsgBox("Invalid file length detected in the following entry:" + vbNewLine + vbNewLine +
                               ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid virtual file length")
                GoTo NoDisk
                'FLN = P.Length - FON
                'FL = ConvertIntToHex(FLN, 4)
            End If

            'Make sure file address+length<=&H10000
            If FAN + FLN > &H10000 Then
                MsgBox("Invalid file address and/or length detected in the following entry:" + vbNewLine + vbNewLine +
                               ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid virtual file address and/or length")
                GoTo NoDisk
                'FLN = &H10000 - FAN
                'FL = ConvertIntToHex(FLN, 4)
            End If

            'Trim file to the specified chunk (FLN number of bytes starting at FON, to Address of FAN)
            Dim PL As List(Of Byte) = P.ToList      'Copy array to list
            P = PL.Skip(FON).Take(FLN).ToArray      'Trim file to specified segment (FLN number of bytes starting at FON)

        Else

            MsgBox("The following file does not exist:" + vbNewLine + vbNewLine + FN, vbOKOnly + vbCritical, "File not found")
            GoTo NoDisk

        End If

        VFileCnt += 1
        ReDim Preserve tmpVFileNameA(VFileCnt), tmpVFileAddrA(VFileCnt), tmpVFileOffsA(VFileCnt), tmpVFileLenA(VFileCnt), tmpVFileIOA(VFileCnt)

        tmpVFileNameA(VFileCnt) = ScriptEntryArray(0) 'FN
        tmpVFileAddrA(VFileCnt) = FA
        tmpVFileOffsA(VFileCnt) = FO     'This may not be needed later
        tmpVFileLenA(VFileCnt) = FL
        tmpVFileIOA(VFileCnt) = FUIO

        tmpVFiles.Add(P)

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")
NoDisk:
        AddVirtualFile = False
    End Function

    Public Function AddFileToBundle() As Boolean
        If DoOnErr Then On Error GoTo Err

        AddFileToBundle = True

        Dim FN As String = ScriptEntryArray(0)
        Dim FA As String = ""
        Dim FO As String = ""
        Dim FL As String = ""
        Dim FAN As Integer
        Dim FON As Integer
        Dim FLN As Integer
        Dim FUIO As Boolean = False

        Dim NumParams As Integer = 1

        Dim P() As Byte

        If Right(FN, 1) = "*" Then
            FN = Replace(FN, "*", "")
            FUIO = True
        End If

        If InStr(FN, ":") = 0 Then  'relative file path
            FN = ScriptPath + FN            'look for file in script's folder
        End If

        'Correct file parameter length to 4-8 characters
        For I As Integer = 1 To ScriptEntryArray.Count - 1

            If ParameterIsNumeric(I) Then
                NumParams += 1
            Else
                Exit For
            End If

            CorrectParameterStringLength(I)

            ''Remove HEX prefix
            'If Left(ScriptEntryArray(I), 1) = "$" Then
            'ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 1)
            'End If
            'Select Case LCase(Left(ScriptEntryArray(I), 2))
            'Case "&h", "0x"
            'ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 2)
            'End Select

            'If Left(ScriptEntryArray(I), 1) = "." Then
            'ScriptEntryArray(I) = ScriptEntryArray(I).TrimStart(".")
            'If IsNumeric(ScriptEntryArray(I)) Then
            'Dim ScriptEntryInt As Integer = Convert.ToInt32(ScriptEntryArray(I))
            'ScriptEntryArray(I) = Hex(ScriptEntryInt)
            'Else
            'Exit For
            'End If
            'End If

            'If IsNumeric("&H" + ScriptEntryArray(I)) Then
            ''If IsHexString(ScriptEntryArray(I)) Then
            'NumParams = I + 1
            'Else
            'Exit For
            'End If

            ''Remove unwanted spaces
            'Replace(ScriptEntryArray(I), " ", "")

            'Select Case I
            'Case 2      'File Offset max. $ffff ffff (dword)
            'If Len(ScriptEntryArray(I)) < 8 Then
            'ScriptEntryArray(I) = Left("00000000", 8 - Len(ScriptEntryArray(I))) +
ScriptEntryArray(I)
            'ElseIf (I = 2) And (Len(ScriptEntryArray(I)) > 8) Then
            'ScriptEntryArray(I) = Right(ScriptEntryArray(I), 8)
            'End If
            'Case Else   'File Address, File Length max. $ffff
            'If Len(ScriptEntryArray(I)) < 4 Then
            'ScriptEntryArray(I) = Left("0000", 4 - Len(ScriptEntryArray(I))) + ScriptEntryArray(I)
            'ElseIf Len(ScriptEntryArray(I)) > 4 Then
            'ScriptEntryArray(I) = Right(ScriptEntryArray(I), 4)
            'End If
            'End Select
        Next

        'Get file variables from script, or get default values if there were none in the script entry
        If IO.File.Exists(FN) = True Then
            P = IO.File.ReadAllBytes(FN)

            Select Case NumParams   'ScriptEntryArray.Count
                Case 1  'No parameters in script
```

```
                    If InStr(LCase(FN), ".sid") <> 0 Then   'SID file - read parameters from file
                        FA = ConvertIntToHex(P(P(7)) + (P(P(7) + 1) * 256), 4)
                        FO = ConvertIntToHex(P(7) + 2, 8)
                        FL = ConvertIntToHex((P.Length - P(7) - 2), 4)
                    Else                                                    'Any other files
                        If P.Length > 2 Then                                'We have at least 3 bytes in the file
                            FA = ConvertIntToHex(P(0) + (P(1) * 256), 4)       'First 2 bytes define load address
                            FO = "00000002"                                    'Offset=2, Length=prg length-2
                            FL = ConvertIntToHex(P.Length - 2, 4)
                        Else                                                'Short file without paramters -> STOP
                            MsgBox("File parameters are needed for the following file:" + vbNewLine + vbNewLine + FN, vbCritical + vbOKOnly, "Missing file parameters")
                            GoTo NoDisk
                        End If
                    End If
                Case 2  'One parameter in script
                    FA = ScriptEntryArray(1)                                'Load address from script
                    FO = "00000000"                                         'Offset will be 0, length=prg length
                    FL = ConvertIntToHex(P.Length, 4)
                Case 3  'Two parameters in script
                    FA = ScriptEntryArray(1)                                'Load address from script
                    FO = ScriptEntryArray(2)                                'Offset from script
                    FON = Convert.ToInt32(FO, 16)                           'Make sure offset is valid
                    If FON > P.Length - 1 Then
                        FON = P.Length - 1                                  'If offset>prg length-1 then correct it
                        FO = ConvertIntToHex(FON, 8)
                    End If                                                  'Length=prg length- offset
                    FL = ConvertIntToHex(P.Length - FON, 4)
                Case 4  'Three parameters in script
                    FA = ScriptEntryArray(1)
                    FO = ScriptEntryArray(2)
                    FON = Convert.ToInt32(FO, 16)                           'Make sure offset is valid
                    If FON > P.Length - 1 Then
                        MsgBox("Invalid offset detected in the following entry:" + vbNewLine + vbNewLine +
                               ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid offset")
                        GoTo NoDisk
                        'FON = P.Length - 1                                  'If offset>prg length-1 then correct it
                        'FO = ConvertIntToHex(FON, 8)
                    End If                                                  'Length=prg length- offset
                    FL = ScriptEntryArray(3)
            End Select

            FAN = Convert.ToInt32(FA, 16)
            FON = Convert.ToInt32(FO, 16)
            FLN = Convert.ToInt32(FL, 16)

            'Make sure file length is not longer than actual file (should not happen)
            If FON + FLN > P.Length Then
                MsgBox("Invalid file length detected in the following entry:" + vbNewLine + vbNewLine +
                               ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid file length")
                GoTo NoDisk
                'FLN = P.Length - FON
                'FL = ConvertIntToHex(FLN, 4)
            End If

            'Make sure file address+length<=&H10000
            If FAN + FLN > &H10000 Then
                MsgBox("Invalid file address and/or length detected in the following entry:" + vbNewLine + vbNewLine +
                               ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid file address and/or length")
                GoTo NoDisk
                'FLN = &H10000 - FAN
                'FL = ConvertIntToHex(FLN, 4)
            End If

            'Trim file to the specified chunk (FLN number of bytes starting at FON, to Address of FAN)
            Dim PL As List(Of Byte) = P.ToList      'Copy array to list
            P = PL.Skip(FON).Take(FLN).ToArray      'Trim file to specified segment (FLN number of bytes starting at FON)

        Else

            MsgBox("The following file does not exist:" + vbNewLine + vbNewLine + FN, vbOKOnly + vbCritical, "File not found")
            GoTo NoDisk

        End If

        FileCnt += 1
        ReDim Preserve tmpFileNameA(FileCnt), tmpFileAddrA(FileCnt), tmpFileOffsA(FileCnt), tmpFileLenA(FileCnt), tmpFileIOA(FileCnt)

        tmpFileNameA(FileCnt) = FN
        tmpFileAddrA(FileCnt) = FA
        tmpFileOffsA(FileCnt) = FO     'This may not be needed later
        tmpFileLenA(FileCnt) = FL
        tmpFileIOA(FileCnt) = FUIO

        UncompBundleSize += Int(FLN / 256)
        If FLN Mod 256 <> 0 Then
            UncompBundleSize += 1
        End If

        If FirstFileOfDisk = True Then      'If Demo Start is not specified, we will use the start address of the first file
            FirstFileStart = FA
            FirstFileOfDisk = False
        End If

        tmpPrgs.Add(P)

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")
NoDisk:
        AddFileToBundle = False

    End Function

    Private Function SplitScriptEntry() As Boolean
        If DoOnErr Then On Error GoTo Err

        SplitScriptEntry = True

        If InStr(ScriptEntry, vbTab) = 0 Then
            ScriptEntryType = Replace(ScriptEntry, " ", "")
            ScriptEntry = ""
        Else
            ScriptEntryType = Replace(Left(ScriptEntry, InStr(ScriptEntry, vbTab) - 1), " ", "")
            ScriptEntry = Right(ScriptEntry, Len(ScriptEntry) - InStr(ScriptEntry, vbTab))
        End If

        LastNonEmpty = -1

        ReDim ScriptEntryArray(LastNonEmpty)

        If ScriptEntry = "" Then Exit Function

        ScriptEntryArray = Split(ScriptEntry, vbTab)

        For I As Integer = 0 To ScriptEntryArray.Length - 1
            If ScriptEntryArray(I) <> "" Then
                LastNonEmpty += 1
                ScriptEntryArray(LastNonEmpty) = ScriptEntryArray(I)
            End If
        Next

        If LastNonEmpty > -1 Then
            ReDim Preserve ScriptEntryArray(LastNonEmpty)
        Else
            ReDim ScriptEntryArray(0)
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

        SplitScriptEntry = False

    End Function

    Public Function ResetDiskVariables() As Boolean
        If DoOnErr Then On Error GoTo Err

        ResetDiskVariables = True

        If DiskCnt = 126 Then
            MsgBox("You have reached the maximum number of disks in this project!", vbOKOnly +
vbInformation, "Can't have more than 127 disks :(")
            GoTo NoDisk
        End If

        DiskCnt += 1
        ReDim Preserve DiskSizeA(DiskCnt)
        'Reset Bundle File variables here, to have an empty array for the first compression on a
ReBuild
        Prgs.Clear()    'this is the one that is needed for the first CompressPart call during a
ReBuild
        ReDim FileNameA(-1), FileAddrA(-1), FileOffsA(-1), FileLenA(-1), FileIOA(-1)    'but reset all
arrays just to be safe

        'Reset directory arrays
        ReDim DirBlocks(511), DirPtr(127)

        'Reset disk system to support 35 tracks
        TracksPerDisk = 35

        'Reset Hi-Score Saver plugin variables
        bSaverPlugin = False
        HSFileName = ""
        HSAddress = 0
        HSOffset = 0
        HSLength = 0

        'Reset interleave
        ResetInterleaves()

        BufferCnt = 0
        BundleNo = 0
        MaxBundleNoExceeded = False

        ReDim ByteSt(-1)
        ResetBuffer()


        D64Name = ""
        DiskHeader = "" '"demo disk " + Year(Now).ToString
        DiskID = "" '"sprkl"
        DemoName = "" '"demo"
        DemoStart = ""
        DirArtName = ""
        LoaderZP = "02"

        'Reset Disk image
        NewDisk()

        BlockPtr = 1

        '-------------------------------------------------------------

        StartTrack = 1 : StartSector = 0

        NextTrack = StartTrack
        NextSector = StartSector

        BitsSaved = 0 : BytesSaved = 0

        FirstFileOfDisk = True  'To save Start Address of first file on disk if Demo Start is not
specified

        '-------------------------------------------------------------

        BundleCnt = -1        'WILL BE INCREASED TO 0 IN ResetPartVariables
        LoaderBundles = 1
        FilesInBuffer = 1

        CurrentBundle = -1

        '-------------------------------------------------------------

        If ResetBundleVariables() = False Then GoTo NoDisk    'Also adds first bundle

        NewBundle = False

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
NoDisk:
        ResetDiskVariables = False

    End Function

    Public Sub ResetInterleaves()
        If DoOnErr Then On Error GoTo Err

        IL0 = DefaultIL0
        IL1 = DefaultIL1
        IL2 = DefaultIL2
        IL3 = DefaultIL3

        Exit Sub

Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
    End Sub

    Public Function ResetBundleVariables() As Boolean
        If DoOnErr Then On Error GoTo Err

        ResetBundleVariables = True

        FileCnt = -1
        ReDim tmpFileNameA(FileCnt), tmpFileAddrA(FileCnt), tmpFileOffsA(FileCnt),
tmpFileLenA(FileCnt), tmpFileIOA(FileCnt)

        VFileCnt = -1
        ReDim tmpVFileNameA(VFileCnt), tmpVFileAddrA(VFileCnt), tmpVFileOffsA(VFileCnt),
tmpVFileLenA(VFileCnt), tmpVFileIOA(VFileCnt)

        tmpVFiles.Clear()

        tmpPrgs.Clear()

        BundleCnt += 1

        TotalBundles += 1
        ReDim Preserve BundleSizeA(TotalBundles), BundleOrigSizeA(TotalBundles)
        BlockCnt = 0

        UncompBundleSize = 0

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

        ResetBundleVariables = False

    End Function

    Public Function AddCompressedBundlesToDisk() As Boolean
        If DoOnErr Then On Error GoTo Err

        AddCompressedBundlesToDisk = True

        If BlocksFree < BufferCnt Then
            MsgBox(D64Name + " cannot be built because it would require " + BufferCnt.ToString + " blocks." + vbNewLine + vbNewLine +
                   "This disk only has " + SectorsPerDisk.ToString + " blocks.", vbOKOnly + vbCritical, "Not enough free space on disk")
            GoTo NoDisk
        End If

        CalcILTab()

        InjectDirBlocks()

        For I = 0 To BufferCnt - 1
            CT = TabT(I)
            CS = TabS(I)
            For J = 0 To 255
                Disk(Track(CT) + 256 * CS + J) = ByteSt(I * 256 + J)
            Next

            DeleteBit(CT, CS, True)
        Next

        If BufferCnt < SectorsPerDisk Then
            NextTrack = TabT(BufferCnt)
            NextSector = TabS(BufferCnt)
        Else
            NextTrack = 18
            NextSector = 0
        End If

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")
NoDisk:
        AddCompressedBundlesToDisk = False

    End Function

    Public Function AddDirArt() As Boolean
        If DoOnErr Then On Error GoTo Err

        AddDirArt = True

        'Make sure strings have values
        If DirArtName Is Nothing Then DirArtName = ""
        If DirArt Is Nothing Then DirArt = ""

        If DirArtName = "" Then Exit Function

        If IO.File.Exists(DirArtName) = False Then
            MsgBox("The following DirArt file does not exist: " + vbNewLine + vbNewLine + DirArtName, vbOKOnly + vbExclamation, "DirArt file cannot be found")
            GoTo NoDisk
        End If

        Dim DAN() As String = DirArtName.Split(".")

        Dim DirArtType As String = ""

        If DAN.Length > 1 Then
```

```
            DirArtType = DAN(DAN.Length - 1)
        End If

        Select Case LCase(DirArtType)
            Case "d64"
                ConvertD64ToDirArt()
            Case "txt"
                ConvertTxtToDirArt()
            Case "prg"
                ConvertBintoDirArt(LCase(DirArtType))
            Case Else
                ConvertBintoDirArt()
        End Select

        Exit Function
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")
NoDisk:
        AddDirArt = False

    End Function

    Private Sub ConvertBintoDirArt(Optional DirArtType As String = "bin")
        If DoOnErr Then On Error GoTo Err

        Dim DA() As Byte = IO.File.ReadAllBytes(DirArtName)

        DirTrack = 18
        DirSector = 1
        Dim NB As Byte = 0
        Dim DAPtr As Integer = If(DirArtType = "prg", 2, 0)
NextSector:
        For B As Integer = DAPtr To DA.Length - 1 Step 40

            FindNextDirPos()

            If DirPos <> 0 Then
                Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 0) = &H82   '"PRG" -  all dir entries will point at first file in dir
                Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 1) = 18     'Track 18 (track pointer of boot loader)
                Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 2) = 7      'Sector 7 (sector pointer of boot loader)

                For I As Integer = 0 To 15
                    If B + I < DA.Length Then
                        Select Case DA(B + I)
                            Case 0 To 31
                                NB = DA(B + I) + 64
                            Case 32 To 63
                                NB = DA(B + I)
                            Case 64 To 95
                                NB = DA(B + I) + 128
                            Case 96 To 127
                                NB = DA(B + I) + 64
                            Case 128 To 159
                                NB = DA(B + I) - 128
                            Case 160 To 191
                                NB = DA(B + I) - 64
                            Case 192 To 223
```

```
                                NB = DA(B + I) - 64
                            Case 224 To 254
                                NB = DA(B + I)
                        End Select
                        Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 3 + I) = NB
                    Else
                        Exit For
                    End If
                Next
                If (DirTrack = 18) AndAlso (DirSector = 1) AndAlso (DirPos = 2) Then
                    'Very first dir entry, also add loader block count
                    Disk(Track(DirTrack) + (DirSector * 256) + DirPos + &H1C) = LoaderBlockCount
                End If
            Else
                Exit For
            End If
        Next

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")

    End Sub

    Private Sub ConvertD64ToDirArt()
        If DoOnErr Then On Error GoTo Err

        Dim DA() As Byte = IO.File.ReadAllBytes(DirArtName)

        Dim T As Integer = 18
        Dim S As Integer = 1

        DirTrack = 18
        DirSector = 1
        Dim DirFull As Boolean = False
NextSector:
        Dim DAPtr As Integer = Track(T) + (S * 256)
        For B As Integer = 2 To 255 Step 32
            If DA(DAPtr + B) <> 0 Then      '= &H82 Then    'PRG file type

                FindNextDirPos()

                If DirPos <> 0 Then
                    Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 0) = &H82   '"PRG" -  all dir entries will point at first file in dir
                    Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 1) = 18     'Track 18 (track pointer of boot loader)
                    Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 2) = 7      'Sector 7 (sector pointer of boot loader)

                    For I As Integer = 0 To 15
                        Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 3 + I) = DA(DAPtr + B + 3 + I)
                    Next

                    If (DirTrack = 18) AndAlso (DirSector = 1) AndAlso (DirPos = 2) Then
                        'Very first dir entry, also add loader block count
                        Disk(Track(DirTrack) + (DirSector * 256) + DirPos + &H1C) = LoaderBlockCount
                    End If
                Else
                    DirFull = True
                    Exit For
                End If
            End If
        Next

        If (DirFull = False) And (DA(DAPtr) <> 0) Then
            T = DA(DAPtr)
            S = DA(DAPtr + 1)
            GoTo NextSector
        End If

        If DiskHeader = "" Then
            For I As Integer = 0 To 15
                Disk(Track(18) + &H90 + I) = DA(Track(18) + &H90 + I)
            Next
        End If

        If DiskID = "" Then
            For I As Integer = 0 To 4
                Disk(Track(18) + &HA2 + I) = DA(Track(18) + &HA2 + I)
            Next
        End If

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Private Sub ConvertTxtToDirArt()
        If DoOnErr Then On Error GoTo Err

        DirArt = IO.File.ReadAllText(DirArtName)

        Dim DirEntries() As String = DirArt.Split(vbLf)

        DirTrack = 18
        DirSector = 1
        For I As Integer = 0 To DirEntries.Count - 1
            DirEntry = DirEntries(I).TrimEnd(Chr(13))
            FindNextDirPos()
            If DirPos <> 0 Then
                AddDirEntry()
            Else
                Exit For
            End If
        Next

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Private Sub AddDirEntry()
        If DoOnErr Then On Error GoTo Err

        Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 0) = &H82   '"PRG" -  all dir entries will
```

```
point at first file in dir
        Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 1) = 18     'Track 18 (track pointer of boot loader)
        Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 2) = 7      'Sector 7 (sector pointer of boot loader)

        'Remove vbNewLine characters and add 16 SHIFT+SPACE tail characters
        DirEntry += StrDup(16, Chr(160))

        'Copy only the first 16 characters of the edited DirEntry to the Disk Directory
        For I As Integer = 1 To 16
            Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 2 + I) = Asc(Mid(UCase(DirEntry), I, 1))
        Next

        If (DirTrack = 18) AndAlso (DirSector = 1) AndAlso (DirPos = 2) Then
            'Very first dir entry, also add loader block count
            Disk(Track(DirTrack) + (DirSector * 256) + DirPos + &H1C) = LoaderBlockCount
        End If

        'Reset DirEntry
        DirEntry = ""

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")

    End Sub

    Public Sub FindNextDirPos()
        If DoOnErr Then On Error GoTo Err

        DirPos = 0

FindNextEntry:

        For I As Integer = 2 To 255 Step 32
            If Disk(Track(DirTrack) + (DirSector * 256) + I) = 0 Then
                DirPos = I
                Exit Sub
            End If
        Next

        FindNextDirSector()

        If DirSector <> 0 Then GoTo FindNextEntry

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + " Error")

    End Sub

    Private Sub FindNextDirSector()
        If DoOnErr Then On Error GoTo Err

        'Sector order: 1,7,13,3,9,15,4,8,12,16

        LastDirSector = DirSector
```

```vb

        If DirSector < 6 Then
            DirSector += 1
        Else
            DirSector = 0
        End If

        'Select Case DirSector
        'Case 1
        'DirSector = 2
        'Case 7
        'DirSector = 13
        'Case 13
        'DirSector = 3
        'Case 3
        'DirSector = 9
        'Case 9
        'DirSector = 15
        'Case 15
        'DirSector = 4
        'Case 4
        'DirSector = 8
        'Case 8
        'DirSector = 12
        'Case 12
        'DirSector = 16
        'Case 16
        'DirSector = 0
        'End Select

        Disk(Track(DirTrack) + (LastDirSector * 256)) = DirTrack
        Disk(Track(DirTrack) + (LastDirSector * 256) + 1) = DirSector

        If DirSector <> 0 Then
            Disk(Track(DirTrack) + (DirSector * 256)) = 0
            Disk(Track(DirTrack) + (DirSector * 256) + 1) = 255
        End If

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Sub SetScriptPath(Path As String)
        If DoOnErr Then On Error GoTo Err

        ScriptName = Path
        ScriptPath = ScriptName

        If Path = "" Then Exit Sub

        For I As Integer = Len(Path) - 1 To 0 Step -1
            If Right(ScriptPath, 1) <> "\" Then
                ScriptPath = Left(ScriptPath, Len(ScriptPath) - 1)
            Else
                Exit For
            End If
        Next

```

```
            Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Sub CalcILTab()
        If DoOnErr Then On Error GoTo Err

        Dim SMax, IL As Integer
        Dim Disk(682 + 85) As Byte
        Dim I As Integer = 0
        Dim SCnt As Integer
        Dim Tr(40) As Integer
        Dim S As Integer = 0
        Dim LastS As Integer = 0

        Tr(1) = 0
        For T = 1 To TracksPerDisk - 1 '34
            Select Case T
                Case 1 To 17
                    Tr(T + 1) = Tr(T) + 21
                Case 18 To 24
                    Tr(T + 1) = Tr(T) + 19
                Case 25 To 30
                    Tr(T + 1) = Tr(T) + 18
                Case 31 To 40
                    Tr(T + 1) = Tr(T) + 17
            End Select
        Next

        For T As Integer = 1 To TracksPerDisk
            TabStartS(T) = 255
            If T = 18 Then
                T += 1
                TabStartS(T) = 255
                S += 2
            End If

            SCnt = 0

            Select Case T
                Case 1 To 17
                    SMax = 21
                    IL = IL0
                Case 18 To 24
                    SMax = 19
                    IL = IL1
                Case 25 To 30
                    SMax = 18
                    IL = IL2
                Case 31 To 40
                    SMax = 17
                    IL = IL3
            End Select

            IL = IL Mod SMax

            'If SectorSkew <> 0 Then
            'If T = 19 Then
            'S = LastS - ((2 * SectorSkew) + 4)      'Extra sector skew for skipping track 18
            'ElseIf T <> 1 Then
            'S = LastS - SectorSkew          'Sector Skew
            'End If
            'If S < 0 Then
            'S += SMax
            'End If
            'Else
            'If T = 18 Then
            'S += 2
            'End If
            'End If

            'S = 0                                  'Reset first sector for each track

            GoTo NextStart

NextSector:
            If Disk(Tr(T) + S) = 0 Then
                Disk(Tr(T) + S) = 1
                TabT(I) = T
                TabS(I) = S
                LastS = S
                TabSCnt(I) = SMax - SCnt
                I += 1
                SCnt += 1
                S += IL
NextStart:
                If S >= SMax Then
                    S -= SMax
                    If (T < 18) And (S > 0) Then S -= 1 'Wrap around: Subtract 1 if S>0 for tracks 1-
17
                    'If S > 0 Then S -= 1                'Wrap around: Subtract 1 if S>0 for all tracks
                End If
                If TabStartS(T) = 255 Then
                    'MsgBox(T.ToString + ":" + S.ToString)
                    TabStartS(T) = S
                End If
                If SCnt < SMax Then GoTo NextSector
            Else
                S += 1
                If S >= SMax Then
                    S = 0
                End If
                If SCnt < SMax Then GoTo NextSector
            End If
        Next

        'IO.File.WriteAllBytes(UserFolder + "\OneDrive\C64\Coding\TabT.bin", TabT)
        'IO.File.WriteAllBytes(UserFolder + "\OneDrive\C64\Coding\TabS.bin", TabS)
        'IO.File.WriteAllBytes(UserFolder + "\OneDrive\C64\Coding\TabStartS.bin", TabStartS)
        'IO.File.WriteAllBytes(UserFolder + "\OneDrive\C64\Coding\TabSCnt.bin", TabSCnt)

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Public Sub GetILfromDisk()
```

```
        If DoOnErr Then On Error GoTo Err

        IL0 = 256 - EORtransform(If(Disk(Track(18) + (0 * 256) + 250) <> 0, Disk(Track(18) + (0 * 256)
+ 250), EORtransform(4)))
        IL1 = 256 - EORtransform(If(Disk(Track(18) + (0 * 256) + 252) <> 0, Disk(Track(18) + (0 * 256)
+ 252), EORtransform(3)))
        IL2 = 256 - EORtransform(If(Disk(Track(18) + (0 * 256) + 253) <> 0, Disk(Track(18) + (0 * 256)
+ 253), EORtransform(3)))
        IL3 = 256 - EORtransform(If(Disk(Track(18) + (0 * 256) + 254) <> 0, Disk(Track(18) + (0 * 256)
+ 254), EORtransform(3)))

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

    End Sub

    Private Sub UpdateBlocksFree()
        If DoOnErr Then On Error GoTo Err

        If TracksPerDisk = ExtTracksPerDisk Then
            Dim ExtBlocksFree As Byte = If(BlocksFree > ExtSectorsPerDisk - StdSectorsPerDisk,
ExtSectorsPerDisk - StdSectorsPerDisk - BlocksUsedBySaver, BlocksFree)
            Disk(Track(18) + 4) += ExtBlocksFree
        End If

        Exit Sub
Err:
        ErrCode = Err.Number
        MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
    End Sub

    'Public Function IsHexString(S As String) As Boolean
    'If DoOnErr Then On Error GoTo Err
    '
    'IsHexString = True

    'For I As Integer = 1 To S.Length
    'Select Case Mid(LCase(S), I, 1)
    'Case " ", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "a", "b", "c", "d", "e", "f"
    'Case Else
    'IsHexString = False
    'Exit For
    'End Select
    'Next

    'Exit Function
    'Err:
    'ErrCode = Err.Number
    'MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + "
Error")

    'End Function

End Module
```