

```

1 Friend Module ModDisk
2     Public DoOnErr As Boolean = True
3     Public ErrCode As Integer = 0
4
5     Public ReadOnly UserDeskTop As String =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop)
6     Public ReadOnly UserFolder As String =
Environment.GetFolderPath(Environment.SpecialFolder.UserProfile)
7
8     Public ReadOnly Ascii2Petscii As Byte() = My.Resources.Ascii2DisplayCode
9
10    Public Enum PackerTypes As Byte
11        Faster = 0
12        Better = 1
13    End Enum
14
15    Public Packer As PackerTypes = PackerTypes.Faster
16
17    'Public DiskLoop As Integer = 0
18
19    Public Drive() As Byte
20    Public DirBlocks(511) As Byte
21    Public DirPtr(127) As Integer
22    Public LastBitPtr As Integer
23    Public LastBufferCnt As Integer
24
25    Public TotLit, TotMatch As Integer
26
27    'Public ReadOnly CustomIL As Boolean = True
28    Public ReadOnly DefaultIL0 As Byte = 4
29    Public ReadOnly DefaultIL1 As Byte = 3
30    Public ReadOnly DefaultIL2 As Byte = 3
31    Public ReadOnly DefaultIL3 As Byte = 3
32    Public IL0 As Byte = DefaultIL0
33    Public IL1 As Byte = DefaultIL1
34    Public IL2 As Byte = DefaultIL2
35    Public IL3 As Byte = DefaultIL3
36
37    Public BufferCnt As Integer = 0
38
39    Public FileUnderIO As Boolean = False
40    Public IOBit As Byte
41
42    Public ReadOnly SectorSkew As Integer = 2          'Skew if disk is formatted using Format II by TLR
43
44    Public ReadOnly StdSectorsPerDisk As Integer = 664          'Standard disk
45    Public ReadOnly StdTracksPerDisk As Integer = 35
46    Public ReadOnly StdBytesPerDisk As Integer = 174848
47
48    Public ReadOnly ExtSectorsPerDisk As Integer = StdSectorsPerDisk + 85          'Exnteded disk
49    Public ReadOnly ExtTracksPerDisk As Integer = 40
50    Public ReadOnly ExtBytesPerDisk As Integer = StdBytesPerDisk + (85 * 256)
51
52    Public SectorsPerDisk As Integer = StdSectorsPerDisk
53    Public TracksPerDisk As Integer = StdTracksPerDisk
54
55    Public BlocksFree As Integer = SectorsPerDisk
56
57    Public TabT(ExtSectorsPerDisk - 1), TabS(ExtSectorsPerDisk - 1), TabSCnt(ExtSectorsPerDisk - 1),
TabStartS(ExtTracksPerDisk) As Byte          'TabStartS is 1 based
58
59    Public Disk(StdBytesPerDisk - 1), NextTrack, NextSector As Byte          'Next Empty Track and Sector

```

```

60 Public MaxSector As Byte = 18, LastSector, Prg(), ReferenceFile() As Byte
61
62 Public Track(40), CT, CS, CP, BlockCnt As Integer
63 Public TestDisk As Boolean = False
64 Public StartTrack As Byte = 1
65 Public StartSector As Byte = 0
66
67 Public ByteSt(), Buffer(255), LastByte, AdLo, AdHi As Byte
68 Public Match, MaxBit, MatchSave(), PrgLen, Distant As Integer
69 Public MatchOffset(), MatchCnt, RLECnt, MatchLen(), MaxSave, MaxOffset, MaxLen, LitCnt, BuffAdd,
PrgAdd As Integer
70 Public MaxSLen, MaxSOff, MaxSSave As Integer
71 Public DistAd(), DistLen(), DistSave(), DistCnt, DistBase As Integer
72 Public DtPos, CmPos, CmLast, DtLen, MatchStart As Integer
73 Public LastPO, LastMS As Integer 'save previous POffset and MatchStart positions to recompress
last block of bundle
74 Public LastBitP, LastBytC, LastBitC As Integer
75 Public MatchType(), MaxType As String
76
77 Public LastByteCt As Integer = 255
78 Public LastMOffset As Integer = 0
79 Public LastMLen As Integer = 0
80 Public LastPOffset As Integer = -1
81 Public LastMType As String = ""
82
83 Public PreMOffset As Integer = 0
84 Public PostMOffset As Integer = 0
85
86 Public BitsSaved As Integer = 0
87 Public BytesSaved As Integer = 0
88
89 Public PreOLMO As Integer = 0
90 Public PostOLMO As Integer = 0
91
92 Public Script As String
93 Public ScriptHeader As String = "[Sparkle Loader Script]"
94 Public ScriptName As String
95 Public DiskNo As Integer
96
97 Public D64Name As String = "" '= My.Computer.FileSystem.SpecialDirectories.MyDocuments
98
99 Public ReadOnly DefaultDiskHeader As String = "demo disk " + Year(Now).ToString
100 Public ReadOnly DefaultDiskID As String = "sprkl"
101 Public ReadOnly DefaultDemoName As String = "demo"
102
103 Public DiskHeader As String = DefaultDiskHeader
104 Public DiskID As String = DefaultDiskID
105 Public DemoName As String = DefaultDemoName
106 Public DemoStart As String = ""
107 Public LoaderZP As String = "02"
108
109 'Hi-Score File variables
110 Public HSFileName As String = ""
111 Public HSFile() As Byte
112 Public HSAddress As Integer = 0
113 Public HSOffset As Integer = 0
114 Public HSLength As Integer = 0
115 Public bSaverPlugin As Boolean = False
116
117 'Product ID - unique to build, same for all disks in build, $000000-$ffffff
118 Public ProductID As Integer = 0
119

```

```

120 'Disk system: 35vs 40 tracks
121 'Public TracksOnDisk As Integer = 35
122
123 Public SystemFile As Boolean = False
124 Public FileChanged As Boolean = False
125
126 Public DiskCnt As Integer = -1
127 Public BundleCnt As Integer = -1
128 Public FileCnt As Integer = -1
129 Public CurrentDisk As Integer = -1
130 Public CurrentBundle As Integer = -1
131 Public CurrentFile As Integer = -1
132 Public CurrentScript As Integer = -1
133 Public BundleNo As Integer = -1
134 Public MaxBundleNoExceeded As Boolean = False
135
136 Public D64NameA(), DiskHeaderA(), DiskIDA(), DemoNameA(), DemoStartA(), DirArtA() As String
137 Public FileNameA(), FileAddrA(), FileOffsA(), FileLenA() As String
138 Public tmpFileNameA(), tmpFileAddrA(), tmpFileOffsA(), tmpFileLenA() As String
139 Public FileIOA() As Boolean
140 Public tmpFileIOA() As Boolean
141 Public BitsNeededForNextBundle As Integer = 0
142
143 Public Prgs As New List(Of Byte())
144 Public tmpPrgs As New List(Of Byte())
145
146 Public VFileCnt As Integer = -1
147 Public VFiles As New List(Of Byte())
148 Public VFileNameA(), VFileAddrA(), VFileOffsA(), VFileLenA() As String
149 Public VFileIOA() As Boolean
150 Public tmpVFiles As New List(Of Byte())
151 Public tmpVFileNameA(), tmpVFileAddrA(), tmpVFileOffsA(), tmpVFileLenA() As String
152 Public tmpVFileIOA() As Boolean
153
154 Public DiskNoA(), DFDiskNoA(), DFBundleNoA(), DiskBundleCntA(), DiskFileCntA() As Integer
155 Public FilesInBundleA() As Integer
156 Public PDiskNoA(), PSizeA() As Integer
157 Public PNewBlockA() As Boolean
158 Public FDiskNoA(), FBundleNoA(), FSizeA() As Integer
159 Public TotalBundles As Integer = 0
160 Public NewFile As String
161
162 Public DiskSizeA() As Integer
163 Public DiskStartBundle() As Integer
164 Public FileSizeA() As Integer
165 Public FBSDisk() As Integer
166 Public BundleBytePtrA() As Integer
167 Public BundleBitPtrA() As Integer
168 Public BundleBitPosA() As Integer
169 Public BundleSizeA() As Integer
170 Public BundleBlockCntA() As Integer
171 Public BundleOrigSizeA() As Integer
172 Public UncompBundleSize As Double = 0
173
174 Public bBuildDisk As Boolean = False
175
176 Public SS, SE, LastSS, LastSE As Integer
177 Public NewBundle As Boolean = False
178 Public ScriptEntryType As String = ""
179 Public ScriptEntry As String = ""
180 Public ScriptLine As String = ""
181 Public ScriptEntryArray() As String

```

```

182 Public LastNonEmpty As Integer = -1
183
184 Public LC(), NM, FM, LM As Integer
185 Public SM1 As Integer = 0
186 Public SM2 As Integer = 0
187 Public OverMaxLit As Boolean = False
188
189 Dim FirstFileOfDisk As Boolean = False
190 Dim FirstFileStart As String = ""
191
192 Public BlockPtr As Integer '= 255
193 Public LastBlockCnt As Byte = 0
194 Public LoaderBundles As Integer = 1
195 Public FilesInBuffer As Byte = 1
196
197 Public TmpSetNewBlock As Boolean = False
198 Public SetNewBlock As Boolean = False      'This will fire at the previous bundle and will set
NewBlock2
199 Public NewBlock As Boolean = False      'This will fire at the specified bundle
200
201 Public DirTrack, DirSector, DirPos As Integer
202 Public DirArt As String = ""
203 Public DirArtName As String = ""
204 Private DirEntry As String = ""
205
206 Private LastDirSector As Byte
207
208 Public ScriptPath As String
209
210 Public CmdLine As Boolean = False
211
212 Private Loader() As Byte
213 Private BlocksUsedBySaver As Integer = 0
214
215 Public CompressBundleFromEditor As Boolean = False
216 Public LastFileOfBundle As Boolean = False
217
218 Public SaverSupportsIO As Boolean = False
219
220 Private LoaderBlockCount As Byte
221
222 Public Sub SetMaxSector()
223     If DoOnErr Then On Error GoTo Err
224
225     Select Case CT
226         Case 1 To 17
227             MaxSector = 20
228             'LastSector = 17
229         Case 18 To 24
230             MaxSector = 18
231             'LastSector = 15
232         Case 25 To 30
233             MaxSector = 17
234             'LastSector = 15
235         Case 31 To 40
236             MaxSector = 16
237             'LastSector = 13
238     End Select
239
240     Exit Sub
241 Err:
242     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name

```

```

+ " Error")
243
244     End Sub
245
246     Public Sub ResetArrays()
247         If DoOnError Then On Error GoTo Err
248
249         DiskCnt = -1
250
251         ReDim DiskNoA(DiskCnt), DiskBundleCntA(DiskCnt), DiskFileCntA(DiskCnt)
252         ReDim D64NameA(DiskCnt), DiskHeaderA(DiskCnt), DiskIDA(DiskCnt), DemoNameA(DiskCnt),
DemoStartA(DiskCnt), DirArtA(DiskCnt)
253         ReDim DiskSizeA(DiskCnt)
254
255         BundleCnt = -1
256         ReDim PDiskNoA(BundleCnt), PSizeA(BundleCnt), BundleSizeA(BundleCnt),
FilesInBundleA(BundleCnt), BundleOrigSizeA(BundleCnt)
257
258         FileCnt = -1
259
260         ReDim FileNameA(FileCnt), DFDiskNoA(FileCnt), DFBundleNoA(FileCnt), FileAddrA(FileCnt),
FileOffsA(FileCnt), FileLenA(FileCnt)
261         ReDim FDiskNoA(FileCnt), FBundleNoA(FileCnt), FSizeA(FileCnt)
262         ReDim FileSizeA(FileCnt), FBSDisk(FileCnt)
263
264         TotalBundles = 0
265
266         Exit Sub
267 Err:
268         ErrCode = Err.Number
269         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
270
271     End Sub
272
273     Public Function FindNextScriptEntry() As Boolean
274         If DoOnError Then On Error GoTo Err
275
276         FindNextScriptEntry = True
277
278 NextLine:
279         If Mid(Script, SS, 1) = Chr(13) Then                                'Check if this is an empty line indicating
a new section
280             NewBundle = True
281             SS += 2                                                            'Skip vbCrLf
282             SE = SS + 1
283             GoTo NextLine
284         ElseIf Mid(Script, SS, 1) = Chr(10) Then                            'Line ends with vbCrLf
285             NewBundle = True
286             SS += 1                                                            'Skip vbCrLf
287             SE = SS + 1
288             GoTo NextLine
289         End If
290
291 NextChar:
292         If (Mid(Script, SE, 1) <> Chr(13)) And (Mid(Script, SE, 1) <> Chr(10)) Then    'Look for vbCrLf
and vbCrLf
293             SE += 1                                                            'Not EOL
294             If SE <= Len(Script) Then                                        'Go to next char if we haven't reached the
end of the script
295                 GoTo NextChar
296             Else
297                 'ScriptEntry = Strings.Mid(Script, SS, SE - SS)              'Reached end of script, finish this

```

```

entry
298         'SS = SE + 2                                'Skip EOL bytes
299         'SE = SS + 1
300         GoTo Done
301     End If
302 Else
303 Done:     ScriptEntry = Mid(Script, SS, SE - SS) 'Found EOL 'Finish this entry
304         ScriptLine = ScriptEntry
305         If Mid(Script, SE, 1) = Chr(13) Then          'Skip vbCrLf (2 chars)
306             SS = SE + 2
307         Else
308             SS = SE + 1                                'Otherwise skip 1 char onyl
309         End If
310         SE = SS + 1
311     End If
312
313     Exit Function
314 Err:
315     ErrCode = Err.Number
316     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
317
318     FindNextScriptEntry = False
319
320 End Function
321
322 Public Sub SplitEntry()
323     If DoOnError Then On Error GoTo Err
324
325     LastNonEmpty = -1
326
327     ReDim ScriptEntryArray(LastNonEmpty)
328
329     ScriptEntryArray = Split(ScriptEntry, vbCrLf)
330
331     'Remove empty strings (e.g. if there are to TABs between entries)
332     For I As Integer = 0 To ScriptEntryArray.Length - 1
333         If ScriptEntryArray(I) <> "" Then
334             LastNonEmpty += 1
335             ScriptEntryArray(LastNonEmpty) = ScriptEntryArray(I)
336         End If
337     Next
338
339     If LastNonEmpty > -1 Then
340         ReDim Preserve ScriptEntryArray(LastNonEmpty)
341     Else
342         ReDim ScriptEntryArray(0)
343     End If
344
345     Exit Sub
346 Err:
347     ErrCode = Err.Number
348     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
349
350 End Sub
351
352 Public Sub UpdateDiskSizeOnTheFly()
353     If DoOnError Then On Error GoTo Err
354
355     CP = Track(18)
356

```

```

357 If TracksPerDisk = ExtTracksPerDisk Then
358     ReDim Preserve Disk(ExtBytesPerDisk - 1)
359     BlocksFree = ExtSectorsPerDisk
360     SectorsPerDisk = ExtSectorsPerDisk
361
362     For Cnt As Integer = (36 + 7) * 4 To ((41 + 7) * 4) - 1
363         Disk(Track(18) + Cnt) = 255
364     Next
365     For Cnt As Integer = 36 To 40
366         Disk(Track(18) + ((Cnt + 7) * 4) + 0) = 17
367         Disk(Track(18) + ((Cnt + 7) * 4) + 3) = 1
368     Next
369 Else
370     ReDim Preserve Disk(StdBytesPerDisk - 1)
371     BlocksFree = StdSectorsPerDisk
372     SectorsPerDisk = StdSectorsPerDisk
373
374     For Cnt As Integer = (36 + 7) * 4 To ((41 + 7) * 4) - 1
375         Disk(Track(18) + Cnt) = 0
376     Next
377     For Cnt As Integer = 36 To 40
378         Disk(Track(18) + ((Cnt + 7) * 4) + 0) = 0
379         Disk(Track(18) + ((Cnt + 7) * 4) + 3) = 0
380     Next
381 End If
382
383 Exit Sub
384 Err:
385     ErrCode = Err.Number
386     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
387
388 End Sub
389
390 Public Sub NewDisk()
391     If DoOnError Then On Error GoTo Err
392
393     If TracksPerDisk = ExtTracksPerDisk Then
394         ReDim Disk(ExtBytesPerDisk - 1)
395         BlocksFree = ExtSectorsPerDisk
396         SectorsPerDisk = ExtSectorsPerDisk
397     Else
398         ReDim Disk(StdBytesPerDisk - 1)
399         BlocksFree = StdSectorsPerDisk
400         SectorsPerDisk = StdSectorsPerDisk
401     End If
402
403     Dim B As Byte
404
405     Dim Cnt As Integer
406
407     CP = Track(18)
408     Disk(CP) = &H12                                'Track#18
409     Disk(CP + 1) = &H1                                'Sector#1
410     Disk(CP + 2) = &H41                                '"A"
411
412     For Cnt = &H90 To &HAA                                'Name, ID, DOS type
413         Disk(CP + Cnt) = &HA0
414     Next
415
416     '-----
417

```

```

418 For Cnt = 1 To Len(DiskHeader)
419     B = Ascii2Petscii(Asc(Mid(DiskHeader, Cnt, 1)))
420     'If B > &H5F Then B -= &H20
421     Disk(CP + &H8F + Cnt) = B
422 Next
423
424 '-----
425
426 For Cnt = 1 To Len(DiskID)          'SPRKL
427     B = Ascii2Petscii(Asc(Mid(DiskID, Cnt, 1)))
428     'If B > &H5F Then B -= &H20
429     Disk(CP + &HA1 + Cnt) = B
430 Next
431
432 '-----
433
434 For Cnt = 4 To (36 * 4) - 1
435     Disk(CP + Cnt) = 255
436 Next
437
438 For Cnt = 1 To 17
439     Disk(CP + (Cnt * 4) + 0) = 21
440     Disk(CP + (Cnt * 4) + 3) = 31
441 Next
442
443 For Cnt = 18 To 24
444     Disk(CP + (Cnt * 4) + 0) = 19
445     Disk(CP + (Cnt * 4) + 3) = 7
446 Next
447
448 For Cnt = 25 To 30
449     Disk(CP + (Cnt * 4) + 0) = 18
450     Disk(CP + (Cnt * 4) + 3) = 3
451 Next
452
453 For Cnt = 31 To 35
454     Disk(CP + (Cnt * 4) + 0) = 17
455     Disk(CP + (Cnt * 4) + 3) = 1
456 Next
457
458 'If TracksPerDisk = ExtTracksPerDisk Then
459 'For Cnt = (36 + 7) * 4 To ((41 + 7) * 4) - 1
460 'Disk(CP + Cnt) = 255
461 'Next
462 'For Cnt = 36 To 40
463 'Disk(CP + ((Cnt + 7) * 4) + 0) = 17
464 'Disk(CP + ((Cnt + 7) * 4) + 3) = 1
465 'Next
466 'End If
467
468 Disk(CP + (18 * 4) + 0) = 17
469 Disk(CP + (18 * 4) + 1) = 252
470
471 CT = 18
472 CS = 1
473
474 SetMaxSector()
475
476 CP = Track(CT) + (256 * CS)
477 Disk(CP + 1) = 255
478
479 NextTrack = 1 : NextSector = 0

```



```

480         Exit Sub
481 Err:
482         ErrCode = Err.Number
483         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
484 + " Error")
485
486     End Sub
487
488     Private Function SectorOK(T As Byte, S As Byte) As Boolean
489         If DoOnError Then On Error GoTo Err
490
491         Dim BP As Integer    'BAM Position for Bit Change
492         Dim BB As Integer    'BAM Bit
493
494         BP = Track(18) + T * 4 + 1 + Int(S / 8)
495         BB = 2 ^ (S Mod 8)    '=0-7
496         If (Disk(BP) And BB) = 0 Then    'Block is already used
497             SectorOK = False
498         Else
499             SectorOK = True            'Block is unused
500         End If
501
502         Exit Function
503 Err:
504         ErrCode = Err.Number
505         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
506 + " Error")
507
508         SectorOK = False
509
510     End Function
511
512     Private Function FindNextFreeSector()
513         If DoOnError Then On Error GoTo Err
514
515         FindNextFreeSector = True
516
517         Dim Counter As Integer = 0
518         Dim MaxS As Integer
519 CheckB:
520         If SectorOK(CT, CS) = False Then
521             CS += 1
522             Counter += 1
523             Select Case CT
524                 Case 1 To 17
525                     MaxS = 21
526                     If CS = 21 Then CS = 0
527                 Case 18 To 24
528                     MaxS = 19
529                     If CS = 19 Then CS = 0
530                 Case 25 To 30
531                     MaxS = 18
532                     If CS = 18 Then CS = 0
533                 Case 31 To 35
534                     MaxS = 17
535                     If CS = 17 Then CS = 0
536             End Select
537             If Counter < MaxS Then
538                 GoTo CheckB
539             Else
540                 FindNextFreeSector = False

```

```

540         End If
541     End If
542
543     Exit Function
544 Err:
545     ErrCode = Err.Number
546     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
547     FindNextFreeSector = False
548
549 End Function
550
551 Public Sub DeleteBit(T As Byte, S As Byte, Optional UpdateFreeBlocks As Boolean = True)
552     If DoOnError Then On Error GoTo Err
553
554     'Ignore tracks > 35
555     'If T > 35 Then Exit Sub
556
557     Dim BP As Integer    'BAM Position for Bit Change
558     Dim BB As Integer    'BAM Bit
559
560     BP = Track(18) + (T * 4) + 1 + Int(S / 8) + If(T > 35, 7 * 4, 0)
561     BB = 255 - (2 ^ (S Mod 8))    '=0-7
562
563     Disk(BP) = Disk(BP) And BB
564
565     BP = Track(18) + (T * 4) + If(T > 35, 7 * 4, 0)
566     Disk(BP) -= 1
567
568     If UpdateFreeBlocks = True Then BlocksFree -= 1
569
570     Exit Sub
571 Err:
572     ErrCode = Err.Number
573     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
574
575 End Sub
576
577 Public Function AddInterleave(Optional IL As Byte = 5) As Boolean
578     If DoOnError Then On Error GoTo Err
579
580     AddInterleave = True
581
582     If TrackIsFull(CT) = True Then 'If this track is full, go to next and check again
583         If (CT = 35) Or (CT = 18) Then            'Reached max track No, disk is full
584             AddInterleave = False
585             Exit Function
586         End If
587         CalcNextSector(IL)
588         CT += 1
589
590         If SystemFile = False Then
591             If CT = 18 Then
592                 CT = 19
593                 CS = 3 'Need to skip 2 sectors while skipping Track 18 (the disk keeps spinning)
594             End If
595         End If
596         'First sector in new track will be #1 and NOT #0!!!
597         'CS = StartSector
598     Else
599         CalcNextSector(IL)

```

```

600         End If
601
602         AddInterleave = FindNextFreeSector()
603
604         Exit Function
605 Err:
606         ErrCode = Err.Number
607         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
608
609         AddInterleave = False
610
611     End Function
612
613     Private Function TrackIsFull(T As Byte) As Boolean
614         If DoOnError Then On Error GoTo Err
615
616         If Disk(Track(18) + T * 4) = 0 Then
617             TrackIsFull = True
618         Else
619             TrackIsFull = False
620         End If
621
622         Exit Function
623 Err:
624         ErrCode = Err.Number
625         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
626
627     End Function
628
629     Private Sub CalcNextSector(Optional IL As Byte = 5)
630         If DoOnError Then On Error GoTo Err
631
632         Select Case CT
633
634             Case 1 To 17      '21 sectors, 0-20
635
636                 If CS > 20 Then
637                     CS -= 21
638                     If CS > 0 Then CS -= 1
639                 End If
640                 CS += IL      'IL=4 always
641                 If CS > 20 Then
642                     CS -= 21
643                     If CS > 0 Then CS -= 1
644                 End If
645
646             Case 18          'Handle Dir Track separately
647                 If CS > 18 Then CS -= 19
648                 CS += IL
649                 If CS > 18 Then CS -= 19
650
651             Case 19 To 24    '19 sectors, 0-18
652                 If CS > 18 Then CS -= 19
653                 CS += IL      'IL=3 always
654                 If CS > 18 Then CS -= 19
655
656             Case 25 To 30    '18 sectors, 0-17
657                 If CS > 17 Then CS -= 18
658                 CS += IL      'IL=3 always
659                 If CS > 17 Then CS -= 18

```

```

660         Case 31 To 35     '17 sectors, 0-16
661             If CS > 16 Then CS -= 17
662             CS += IL       'IL=3 always
663             If CS > 16 Then CS -= 17
664
665
666     End Select
667
668     Exit Sub
669 Err:
670     ErrCode = Err.Number
671     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
672
673 End Sub
674
675 Public Function EORtransform(Input As Byte) As Byte
676
677     EORtransform = Input
678
679     Select Case (Input And &H9)
680         Case 0, 9
681             Return EORtransform Xor &H7F
682         Case 1, 8
683             Return EORtransform Xor &H76
684     End Select
685
686     'Select Case (Input And &H99)
687     'Case 0, 9, &H90, &H99
688     'Return EORtransform Xor &HFF
689     'Case 1, 8, &H91, &H98
690     'Return EORtransform Xor &HF6
691     'Case &H10, &H19, &H80, &H89
692     'Return EORtransform Xor &H6F
693     'Case &H11, &H18, &H81, &H88
694     'Return EORtransform Xor &H66
695     'End Select
696
697     'Select Case (Input And &H69)
698     'Case 0, 9, &H60, &H69
699     'Return EORtransform Xor &HFF
700     'Case 1, 8, &H61, &H68
701     'Return EORtransform Xor &HF6
702     'Case &H20, &H29, &H40, &H49
703     'Return EORtransform Xor &H9F
704     'Case &H21, &H28, &H41, &H48
705     'Return EORtransform Xor &H96
706     'End Select
707
708
709 End Function
710
711 Public Sub InjectDirBlocks()
712
713     'DirBlocks(0) = EORtransform(Track)
714     'DirBlocks(1) = EORtransform(Sector)
715     'DirBlocks(2) = EORtransform(Remaining sectors on track)
716     'DirBlocks(3) = BitPtr
717
718     If BundleNo >= 0 Then      'This may be unnecessary, there is always at least 1 bundle on the
disk
719         For I As Integer = BundleNo + 1 To 127

```

```

720         DirBlocks((I * 4) + 3) = DirBlocks((BundleNo * 4) + 3)
721         DirPtr(I) = DirPtr(BundleNo)
722     Next
723 End If
724
725 For I As Integer = 0 To 127
726     DirBlocks(I * 4) = EORtransform(TabT(DirPtr(I)))
727     DirBlocks((I * 4) + 1) = EORtransform(TabStartS(TabT(DirPtr(I))))
728     DirBlocks((I * 4) + 2) = EORtransform(TabSCnt(DirPtr(I)))
729 Next
730
731 'Resort directory sectors to allow simple copy from $0100 to $0700
732 'Dir Block: $00,$ff,$fe,$fd,$fc,...,$01
733 'Buffer:    $00,$01,$02,$03,$04,...,$ff
734
735 Dim DB0(255), DB1(255) As Byte
736 Dim B As Integer = 0          '255 for SD2
737 For I As Integer = 0 To 255
738     DB0(B) = DirBlocks(I)
739     DB1(B) = DirBlocks(I + 256)
740     B -= 1
741     If B < 0 Then B += 256
742 Next
743
744 For I = 0 To 255
745     DirBlocks(I) = DB0(I)
746     DirBlocks(I + 256) = DB1(I)
747 Next
748
749 For I As Integer = 0 To 511
750     Disk(Track(18) + (17 * 256) + I) = DirBlocks(I)
751 Next
752
753 End Sub
754
755 Public Function InjectDriveCode(idcDiskID As Byte, IDCFileCnt As Byte, IDCNextID As Byte, Optional
TestDisk As Boolean = False) As Boolean
756     If DoOnError Then On Error GoTo Err
757
758     InjectDriveCode = True
759
760     Dim I, Cnt As Integer
761
762     'If TestDisk = True Then
763     'Drive = My.Resources.SDT
764     'Else
765     Drive = My.Resources.SD
766     ReDim Preserve Drive((6 * 256) + 1)
767     'End If
768
769     Dim B3(255) As Byte
770     Dim B As Integer = 0          '255 for SD2
771
772     'Resort and EOR transform Block 3
773     For I = 0 To 255
774         B3(B) = EORtransform(Drive((3 * 256) + I + 2))
775         B -= 1
776         If B < 0 Then B += 256
777     Next
778     '-----
779     '    VersionInfo
780     '-----

```

```

781      'Add version info: YY MM DD VV
782      Dim VI As Integer = &H5B
783      Drive(VI + 0 + 2) = (Int(My.Application.Info.Version.Build / 10) * 16) +
(My.Application.Info.Version.Build Mod 10)
784      Drive(VI + 1 + 2) = (Int(My.Application.Info.Version.Revision / 1000) * 16) +
(Int(My.Application.Info.Version.Revision / 100) Mod 10)
785      Drive(VI + 2 + 2) = ((Int(My.Application.Info.Version.Revision / 10) Mod 10) * 16) +
(My.Application.Info.Version.Revision Mod 10)
786      Drive(VI + 4 + 2) = (My.Application.Info.Version.Major * 16) +
My.Application.Info.Version.Minor
787
788      '-----
789      '   ProductID
790      '-----
791      'Add Product ID (add 2 to address for PRG header)
792      Dim PID As Integer = &H1B
793      Drive(PID + 0 + 2) = Int(ProductID / &H10000) And &HFF
794      Drive(PID + 1 + 2) = Int(ProductID / &H100) And &HFF
795      Drive(PID + 2 + 2) = ProductID And &HFF
796
797      '-----
798      '   NoFlipTab
799      '-----
800      'Save last, "dummy" bundle info, needs REVERSED EOR Transform as it is used in the drive code
(add 2 to address for PRG header)
801      'Dim NFT As Integer = &H21
802      'Drive(NFT + 0 + 2) = TabT(LastBufferCnt)
803      'Drive(NFT + 1 + 2) = TabStartS(TabT(LastBufferCnt))
804      'Drive(NFT + 2 + 2) = TabSCnt(LastBufferCnt)
805      'Drive(NFT + 3 + 2) = EORtransform(LastBitPtr)
806
807      'Resort blocks in drive code:
808      For I = 0 To 255
809          Drive((3 * 256) + I + 2) = Drive((4 * 256) + I + 2)      'Copy ZP GCR Tab and GCR loop to
block 3 for loading
810          Drive((4 * 256) + I + 2) = Drive((5 * 256) + I + 2)      'Copy Init code to block 4 for
loading
811          Drive((5 * 256) + I + 2) = B3(I)                          'Copy original block 3 EOR
transformed to block 5 to be loaded by init code
812      Next
813
814      CT = 18
815      CS = 11
816      For Cnt = 0 To 5      '6 blocks to be saved: 18:11, 18:12, 18:13, 18:14, 18:15, (18:16 -
block 5)
817          For I = 0 To 255
818              If Drive.Length > Cnt * 256 + I + 2 Then
819                  Disk(Track(CT) + CS * 256 + I) = Drive(Cnt * 256 + I + 2)
820              End If
821          Next
822          DeleteBit(CT, CS, False)
823          CS += 1
824      Next
825
826      'Next Side Info on last 2 bytes of BAM!!! (Buffer address: $0101-$0102)
827      Disk(Track(18) + (0 * 256) + 255) = EORtransform(idcDiskID)
828      Disk(Track(18) + (0 * 256) + 251) = EORtransform(idcNextID)
829
830      'Add Custom Interleave Info (Buffer address: $0103-$0107)
831      Disk(Track(18) + (0 * 256) + 254) = EORtransform(256 - IL3)
832      Disk(Track(18) + (0 * 256) + 253) = EORtransform(256 - IL2)
833      Disk(Track(18) + (0 * 256) + 252) = EORtransform(256 - IL1)
834      'Disk(Track(18) + (0 * 256) + 250) = EORtransform(IL0)

```

```

835 Disk(Track(18) + (0 * 256) + 250) = EORtransform(256 - IL0)
836
837 "'Dummy" bundle info EOR transformed - to be copied to NoFlipTab after disk flip (Buffer
address: $0108-$010b)
838 'Disk(Track(18) + (0 * 256) + 248) = EORtransform(TabT(LastBufferCnt))
839 'Disk(Track(18) + (0 * 256) + 247) = EORtransform(TabStartS(TabT(LastBufferCnt)))
840 'Disk(Track(18) + (0 * 256) + 246) = EORtransform(TabSCnt(LastBufferCnt))
841 'Disk(Track(18) + (0 * 256) + 245) = LastBitPtr
842
843 'Add IncludeSaveCode flag (Buffer address: $010c)
844 BlocksUsedBySaver = 0
845 If bSaverPlugin Then
846     Disk(Track(18) + (0 * 256) + 249) = EORtransform(2)
847     InjectSaverPlugin()
848 Else
849     Disk(Track(18) + (0 * 256) + 249) = EORtransform(0)
850 End If
851
852 'Also add Product ID to BAM, EOR-transformed (Buffer address: $010d-$010f)
853 Disk(Track(18) + (0 * 256) + 248) = EORtransform(Int(ProductID / &H10000) And &HFF)
854 Disk(Track(18) + (0 * 256) + 247) = EORtransform(Int(ProductID / &H100) And &HFF)
855 Disk(Track(18) + (0 * 256) + 246) = EORtransform(ProductID And &HFF)
856
857 'Add NextID and IL0-IL3 to ZPTab (could be done before Drive code is injected)
858 Dim ZPNextIDLoc As Integer = &H60
859
860 Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 0) = 256 - IL3
861 Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 1) = 256 - IL2
862 Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 2) = 256 - IL1
863 Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 3) = idcNextID
864 Disk(Track(18) + (14 * 256) + ZPNextIDLoc + 4) = 256 - IL0
865
866 Exit Function
867 Err:
868     ErrCode = Err.Number
869     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
870     Resume
871     InjectDriveCode = False
872
873 End Function
874
875 Private Sub InjectSaverPlugin()
876     If DoOnErr Then On Error GoTo Err
877
878     If bSaverPlugin = False Then Exit Sub
879     If HSFile.Length = 0 Then Exit Sub
880     If HSFileName = "" Then Exit Sub
881     If BundleNo > 125 Then
882         MsgBox("The Hi-Score File Saver Plugin cannot be added to the disk because the number of
file bundles exceeds 126!" + vbNewLine + vbNewLine +
883             "The Plugin and the Hi-Score File would use bundle indices $7e and $7f, respectively.",
vbOKOnly + vbExclamation, "Hi-Score File Saver Plugin Error")
884         Exit Sub
885     End If
886
887     BlocksUsedBySaver = Int(HSLength / &H100) + 1 + 2
888
889     If BlocksFree < BlocksUsedBySaver Then
890         MsgBox("The Hi-Score File Saver Plugin cannot be added because there is not enough free
space on the disk!" + vbNewLine + vbNewLine +
891             "The Plugin and the Hi-Score File would need " + BlocksUsedBySaver.ToString + " free
blocks but there " + If(BlocksFree = 1, "is", "are") + " only " + BlocksFree.ToString +

```

```

892     " block" If(BlocksFree = 1, "", "s") + " available on the disk.", vbOKOnly +
vbExclamation, "Hi-Score File Saver Plugin Error")
893     Exit Sub
894 End If
895
896 If InStr(HSFileName, "*") <> 0 Then
897     SaverSupportsIO = True
898     Replace(HSFileName, "*", "")
899 Else
900     SaverSupportsIO = False
901 End If
902
903 Dim SaveCode() As Byte
904
905 If SaverSupportsIO Then
906     SaveCode = My.Resources.SSI0
907 Else
908     SaveCode = My.Resources.SS
909 End If
910
911 'UpdateZP BUG REPORTED BY Rico/Pretzel Logic
912 'WE ALSO NEED TO UPDATE ZP OFFSET IN THE SAVER CODE!!!
913
914 'Convert LoaderZP to byte - it has already been validated in UpdateZP
915 Dim ZP As Byte = Convert.ToByte(LoaderZP, 16)
916
917 If ZP <> 2 Then
918     Dim OPC_STAZP As Byte = &H85 'ZP, ZP+1, Bits
919     Dim OPC_LDAZPY As Byte = &HB1 'ZP
920
921     Dim ZPBase As Byte = &H2
922
923     For I As Integer = 0 To 249
924         If (SaveCode(I) = OPC_STAZP) Or (SaveCode(I) = OPC_LDAZPY) Then
925             If SaveCode(I + 1) = ZPBase Then
926                 SaveCode(I + 1) = ZP
927                 I += 1
928             End If
929         End If
930     Next
931
932     For I As Integer = 0 To 249
933         If (SaveCode(I) = OPC_STAZP) And (SaveCode(I + 1) = ZPBase + 1) Then
934             SaveCode(I + 1) = ZP + 1
935             I += 1
936         End If
937     Next
938
939     For I As Integer = 0 To 249
940         If (SaveCode(I) = OPC_STAZP) And (SaveCode(I + 1) = ZPBase + 2) Then
941             SaveCode(I + 1) = ZP + 2
942             I += 1
943         End If
944     Next
945
946 End If
947
948 SaveCode(2 + &H3) = Int(HSLength / &H100) + 1 'Add 2 for PRG offset!
949 SaveCode(2 + &H13) = (HSAddress - 1) And &HFF
950 SaveCode(2 + &H1A) = Int((HSAddress - 1) / &H100)
951
952 'For I As Integer = 0 To SaveCode.Count - 3

```



```

953      'Find JSR $01e5 (JSR Set01 - expected to remain constant)
954      'If SaveCode(I) = &H20 And SaveCode(I + 1) = &HE5 And SaveCode(I + 2) = &H1 Then
955      'SaveCode(I - 11) = (HSAddress - 1) And &HFF
956      'SaveCode(I - 4) = Int((HSAddress - 1) / &H100)
957      'Exit For
958      'End If
959      'Next
960
961      'Calculate sector pointer on disk
962      Dim SctPtr As Integer = SectorsPerDisk - 2 - (Int(HSLength / 256) + 1)
963
964      'Identify first T/S of the saver plugin
965      CT = TabT(SctPtr)
966      CS = TabS(SctPtr)
967
968      'Copy first block of saver plugin to disk
969      For I As Integer = 0 To 255
970          Disk(Track(CT) + (CS * 256) + I) = SaveCode(2 + I)
971      Next
972
973      'Mark sector off in BAM
974      DeleteBit(CT, CS, True)
975
976      'Add plugin to directory
977      Disk(Track(18) + (18 * 256) + 8) = EORtransform(CT)          'DirBlocks(0) =
EORtransform(Track) = 35
978      Disk(Track(18) + (18 * 256) + 7) = EORtransform(TabStartS(CT)) 'DirBlocks(1) =
EORtransform(Sector) = First sector of Track(35) (not first sector of file!!!)
979      Disk(Track(18) + (18 * 256) + 6) = EORtransform(TabSCnt(SctPtr)) 'DirBlocks(2) =
EORtransform(Remaining sectors on track)
980      Disk(Track(18) + (18 * 256) + 5) = &HFE                      'DirBlocks(3) = BitPtr
981
982      'Next Sector
983      SctPtr += 1
984
985      'Second T/S of saver plugin
986      CT = TabT(SctPtr)
987      CS = TabS(SctPtr)
988
989      'Copy second block of saver plugin to disk
990      For I As Integer = 0 To SaveCode.Length - 256 - 1 - 2
991          Dim J As Integer = 0 - I
992          If J < 0 Then J += 256
993          Disk(Track(CT) + (CS * 256) + J) = EORtransform(SaveCode(256 + 2 + I))
994      Next
995
996      'Mark sector off in BAM
997      DeleteBit(CT, CS, True)
998
999      'Add SaveFile
1000     SctPtr += 1
1001
1002     CT = TabT(SctPtr)
1003     CS = TabS(SctPtr)
1004
1005     Disk(Track(18) + (18 * 256) + 4) = EORtransform(CT)          'DirBlocks(0) =
EORtransform(Track) = 35
1006     Disk(Track(18) + (18 * 256) + 3) = EORtransform(TabStartS(CT)) 'DirBlocks(1) =
EORtransform(Sector) = First sector of Track(35) (not first sector of file!!!)
1007     Disk(Track(18) + (18 * 256) + 2) = EORtransform(TabSCnt(SctPtr)) 'DirBlocks(2) =
EORtransform(Remaining sectors on track)
1008     Disk(Track(18) + (18 * 256) + 1) = &HFE                      'DirBlocks(3) = BitPtr
1009

```

```

1010 DeleteBit(CT, CS, True)
1011
1012 Dim Buffer(255) As Byte
1013 Dim HSStartAdd As Integer = HSAddress + HSLength - 1
1014 Dim BlockCnt = Int(HSLength / 256)
1015
1016 'First block
1017 Buffer(0) = 0
1018 Buffer(1) = EORtransform(Int(HSLength / 256)) 'Remaining block count (EOR
transformed)
1019 Buffer(255) = &HFE 'First byte of block
1020 Buffer(254) = &H81 'Bit stream
1021 Buffer(253) = HSStartAdd Mod 256 'Last byte's address (Lo)
1022 If SaverSupportsIO Then
1023     Buffer(252) = 0 'I/O flag
1024     Buffer(251) = Int(HSStartAdd / 256) 'Last byte's address (Hi)
1025     Buffer(250) = 0 'LongLit flag
1026     Buffer(249) = &HF6 'Number of literals - 1
1027 Else
1028     Buffer(252) = Int(HSStartAdd / 256) 'Last byte's address (Hi)
1029     Buffer(251) = 0 'LongLit flag
1030     Buffer(250) = &HF7 'Number of literals - 1
1031 End If
1032
1033 For I As Integer = 2 To If(SaverSupportsIO, 248, 249)
1034     Buffer(I) = HSFile(HSLength - 1 - If(SaverSupportsIO, 248, 249) + I)
1035 Next
1036
1037 For I As Integer = 0 To 255
1038     Disk(Track(CT) + CS * 256 + I) = Buffer(I)
1039 Next
1040
1041 If SaverSupportsIO Then
1042     HSStartAdd -= &HF7
1043     HSLength -= &HF7
1044 Else
1045     HSStartAdd -= &HF8
1046     HSLength -= &HF8
1047 End If
1048
1049 'Blocks 1 to BlockCnt-1
1050 For I As Integer = 1 To BlockCnt - 1
1051
1052     SctPtr += 1
1053
1054     CT = TabT(SctPtr)
1055     CS = TabS(SctPtr)
1056
1057     DeleteBit(CT, CS, True)
1058
1059     ReDim Buffer(255)
1060
1061     Buffer(0) = &H81 'Bit stream
1062     Buffer(255) = HSStartAdd Mod 256 'Last byte's address (Lo)
1063     If SaverSupportsIO Then
1064         Buffer(254) = 0 'I/O flag
1065         Buffer(253) = Int(HSStartAdd / 256) 'Last byte's address (hi)
1066         Buffer(252) = 0 'LongLit flag
1067         Buffer(251) = &HF9 'Number of literals - 1
1068     Else
1069         Buffer(254) = Int(HSStartAdd / 256) 'Last byte's address (hi)
1070         Buffer(253) = 0 'LongLit flag

```

```

1071         Buffer(252) = &HFA                                     'Number of literals - 1
1072     End If
1073
1074     For J As Integer = 1 To If(SaverSupportsIO, 250, 251)
1075         Buffer(J) = HSFile(HSLength - 1 - If(SaverSupportsIO, 250, 251) + J)
1076     Next
1077
1078     For J As Integer = 0 To 255
1079         Disk(Track(CT) + (CS * 256) + J) = Buffer(J)
1080     Next
1081
1082     If SaverSupportsIO Then
1083         HSStartAdd -= &HFA
1084         HSLength -= &HFA
1085     Else
1086         HSStartAdd -= &HFB
1087         HSLength -= &HFB
1088     End If
1089
1090 Next
1091
1092 'Last block of Hi-Score File
1093 SctPtr += 1
1094
1095 CT = TabT(SctPtr)
1096 CS = TabS(SctPtr)
1097
1098 DeleteBit(CT, CS, True)
1099
1100 ReDim Buffer(255)
1101
1102 Buffer(0) = &H81                                               'Bit stream
1103 Buffer(1) = EORtransform(0)                                     'New block count = 0 (eor transformed)
1104 Buffer(255) = HSStartAdd Mod 256                                'Last byte's address (Lo)
1105 If SaverSupportsIO Then
1106     Buffer(254) = 0                                             'I/O flag
1107     Buffer(253) = Int(HSStartAdd / 256)                         'Last byte's address (Hi)
1108     Buffer(252) = 0                                             'Longlit flag
1109     Buffer(251) = HSLength - 1                                   'Number of remaining literals - 1
1110 Else
1111     Buffer(254) = Int(HSStartAdd / 256)                         'Last byte's address (Hi)
1112     Buffer(253) = 0                                             'Longlit flag
1113     Buffer(252) = HSLength - 1                                   'Number of remaining literals - 1
1114 End If
1115
1116 For I As Integer = 0 To HSLength - 1
1117     Buffer(If(SaverSupportsIO, 251, 252) - HSLength + I) = HSFile(I)
1118 Next
1119
1120 Buffer(If(SaverSupportsIO, 251, 252) - HSLength - 1) = &HF8    'End of File
1121 Bundle flag
1122
1123 For I As Integer = 0 To 255
1124     Disk(Track(CT) + (CS * 256) + I) = Buffer(I)
1125 Next
1126
1127 Exit Sub
1128 Err:
1129     ErrCode = Err.Number
1130     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")

```

```

1131 End Sub
1132
1133 Public Function InjectLoader(DiskIndex As Integer, T As Byte, S As Byte, IL As Byte, Optional
TestDisk As Boolean = False) As Boolean
1134     If DoOnError Then On Error GoTo Err
1135
1136     InjectLoader = True
1137
1138     Dim B, I, Cnt, W As Integer
1139     Dim ST, SS, A, AdLo, AdHi As Byte
1140
1141     'Check if we have a Demo Start Address
1142     If DiskIndex > -1 Then
1143         If DemoStartA(DiskIndex) <> "" Then B = Convert.ToInt32(DemoStartA(DiskIndex), 16)
1144     Else
1145         If DemoStart <> "" Then B = Convert.ToInt32(DemoStart, 16)
1146     End If
1147
1148     'No Demo Start Address, check if we have the first file's start address
1149     If B = 0 Then
1150         If FirstFileStart <> "" Then B = Convert.ToInt32(FirstFileStart, 16)
1151     End If
1152
1153     If B = 0 Then
1154         MsgBox("Unable to build demo disk." + vbNewLine + vbNewLine + "Missing start address",
vbOKOnly)
1155         InjectLoader = False
1156         Exit Function
1157     End If
1158
1159     AdLo = (B - 1) Mod 256
1160     AdHi = Int((B - 1) / 256)
1161
1162     If TestDisk = False Then
1163         Loader = My.Resources.SL
1164         UpdateZP()
1165     Else
1166         Loader = My.Resources.SLT
1167     End If
1168
1169     For I = 0 To Loader.Length - 6 'Find JMP Sparkle_LoadFetched instruction
1170         If (Loader(I) = &H10) And (Loader(I + 3) = &HAD) And (Loader(I + 5) = &H4C) Then
1171             Loader(I) = AdHi 'Hi Byte return address at the end of Loader
1172             Loader(I + 3) = AdLo 'Lo Byte return address at the end of Loader
1173             Exit For
1174         End If
1175     Next
1176
1177     'Number of blocks in Loader
1178     LoaderBlockCount = Int(Loader.Length / 254)
1179     If (Loader.Length) Mod 254 <> 0 Then
1180         LoaderBlockCount += 1
1181     End If
1182
1183     CT = T
1184     CS = S
1185
1186     For I = 0 To LoaderBlockCount - 1
1187         ST = CT
1188         SS = CS
1189         For Cnt = 0 To 253
1190             If (I * 254) + Cnt < Loader.Length Then

```

```

1191         Disk(Track(CT) + (CS * 256) + 2 + Cnt) = Loader((I * 254) + Cnt)
1192     End If
1193 Next
1194 DeleteBit(CT, CS, False)
1195
1196 AddInterleave(IL) 'Go to next free sector with Interleave IL
1197 If I < LoaderBlockCount - 1 Then
1198     Disk(Track(ST) + (SS * 256) + 0) = CT
1199     Disk(Track(ST) + (SS * 256) + 1) = CS
1200 Else
1201     Disk(Track(ST) + (SS * 256) + 0) = 0
1202     If Loader.Length Mod 254 = 0 Then
1203         Disk(Track(ST) + (SS * 256) + 1) = 254 + 1
1204     Else
1205         Disk(Track(ST) + (SS * 256) + 1) = ((Loader.Length) Mod 254) + 1
1206     End If
1207 End If
1208 Next
1209
1210 'AddDemoNameToDisk(DiskIndex, T, S)
1211
1212 Exit Function
1213 Err:
1214     ErrCode = Err.Number
1215     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
1216
1217     InjectLoader = False
1218
1219 End Function
1220
1221 Private Sub AddDemoNameToDisk(DiskIndex As Integer, T As Byte, S As Byte)
1222     If DoOnError Then On Error GoTo Err
1223
1224     Dim B, Cnt As Integer
1225     Dim DN As String = ""
1226     Dim A As Byte
1227
1228     If DiskIndex > -1 Then
1229         DN = If(DemoNameA(DiskIndex) <> "", DemoNameA(DiskIndex), "")
1230     Else
1231         DN = If(DemoName <> "", DemoName, "")
1232     End If
1233
1234     If DN = "" Then
1235         'No DemoName defined, check if we have a DirArt file attached
1236         If DirArtName <> "" Then
1237             'Dirart attached, we will add first dir entry there
1238             Exit Sub
1239         Else
1240             'No DirArt - we need a default dir entry
1241             'DemoName = DefaultDemoName
1242         End If
1243     End If
1244
1245     CT = 18 : CS = 1
1246     Cnt = Track(CT) + (CS * 256)
1247 SeekNewDirBlock:
1248     If Disk(Cnt) <> 0 Then
1249         Cnt = Track(Disk(Cnt)) + Disk(Cnt + 1) * 256
1250         GoTo SeekNewDirBlock
1251     Else

```

```

1252         B = 2
1253 SeekNewEntry:
1254     If Disk(Cnt + B) = &H0 Then
1255
1256         Disk(Cnt + B) = &H82
1257         Disk(Cnt + B + 1) = T
1258         Disk(Cnt + B + 2) = S
1259         For W = 0 To 15
1260             Disk(Cnt + B + 3 + W) = &HA0
1261         Next
1262
1263         For W = 0 To Len(DN) - 1
1264             A = Ascii2PetSCII(Asc(Mid(DN, W + 1, 1)))
1265             'If A > &H5F Then A -= &H20
1266             Disk(Cnt + B + 3 + W) = A
1267         Next
1268         Disk(Cnt + B + &H1C) = LoaderBlockCount      'Length of boot loader in blocks
1269     Else
1270         B += 32
1271         If B < 256 Then
1272             GoTo SeekNewEntry
1273         Else
1274             CS += 4
1275             If CS > 18 Then S -= 18
1276             Disk(Cnt) = CT
1277             Disk(Cnt + 1) = CS
1278             Cnt = Track(CT) + CS * 256
1279             Disk(Cnt) = 0
1280             Disk(Cnt + 1) = 255
1281             GoTo SeekNewDirBlock
1282         End If
1283     End If
1284 End If
1285
1286 Exit Sub
1287 Err:
1288     ErrCode = Err.Number
1289     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
1290
1291 End Sub
1292 Private Sub UpdateZP()
1293     If DoOnError Then On Error GoTo Err
1294
1295     'Check string length
1296     If LoaderZP.Length < 2 Then
1297         LoaderZP = Left("02", 2 - LoaderZP.Length) + LoaderZP
1298     ElseIf LoaderZP.Length > 2 Then
1299         LoaderZP = Right(LoaderZP, 2)
1300     End If
1301
1302     'Convert LoaderZP to byte
1303     Dim ZP As Byte = Convert.ToByte(LoaderZP, 16)
1304
1305     'ZP cannot be $00, $01, or $ff
1306     If ZP < 2 Then
1307         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)
1308         ZP = 2
1309         LoaderZP = "02"
1310     End If
1311     If ZP > &HFD Then

```

```

1312 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)
1313 ZP = &HFD
1314 LoaderZP = "fd"
1315 End If
1316
1317 'ZP=02 is the default, no need to update
1318 If ZP = 2 Then Exit Sub
1319
1320 'ZPUpdate BUG REPORTED BY Rico/Pretzel Logic
1321
1322 'Find the JMP $0700 sequence in the code to identify the beginning of loader
1323 Dim LoaderBase As Integer = &HFFFF
1324 For I As Integer = 0 To Loader.Length - 1 - 2
1325     If (Loader(I) = &H4C) AndAlso (Loader(I + 1) = &H0) AndAlso (Loader(I + 2) = &H7) Then
1326         LoaderBase = I + 3
1327     Exit For
1328 End If
1329 Next
1330
1331 If LoaderBase = &HFFFF Then
1332     MsgBox("Zeropage offset could not updated. Sparkle will use the default zeropage offset
value of $02.", vbInformation + vbOKOnly, "Error updating zeropage offset")
1333     Exit Sub
1334 End If
1335
1336 Dim OPC_STAZP As Byte = &H85
1337 Dim OPC_ADCZP As Byte = &H65
1338 Dim OPC_STAZPY As Byte = &H91
1339 Dim OPC_DECZP As Byte = &HC6
1340 Dim OPC_LDAZP As Byte = &HA5
1341 Dim OPC_RORZP As Byte = &H66
1342 Dim OPC_AS LZP As Byte = &H6
1343 Dim OPC_EORIMM As Byte = &H49
1344 'Dim OPC_LDAZPY As Byte = &HB1
1345
1346 Dim ZPBase As Byte = &H2
1347
1348 For I As Integer = LoaderBase To Loader.Length - 1 - 1
1349     If (Loader(I) = OPC_STAZP) Or
1350         (Loader(I) = OPC_ADCZP) Or
1351         (Loader(I) = OPC_STAZPY) Then
1352
1353         If (Loader(I + 1) = ZPBase) And (Loader(I + 2) <> OPC_EORIMM) Then 'Skip STA $0265
EOR #$FF
1354             Loader(I + 1) = ZP
1355             I += 1
1356         End If
1357     End If
1358 Next
1359
1360 For I As Integer = LoaderBase To Loader.Length - 1 - 1
1361     If (Loader(I) = OPC_STAZP) Or
1362         (Loader(I) = OPC_DECZP) Or
1363         (Loader(I) = OPC_LDAZP) Then
1364
1365         If Loader(I + 1) = ZPBase + 1 Then
1366             Loader(I + 1) = ZP + 1
1367             I += 1
1368         End If
1369     End If
1370 Next

```

```

1371 For I As Integer = LoaderBase To Loader.Length - 1 - 1
1372     If (Loader(I) = OPC_STAZP) Or
1373         (Loader(I) = OPC_RORZP) Or
1374         (Loader(I) = OPC_AS LZP) Then
1375
1376         If Loader(I + 1) = ZPBase + 2 Then
1377             Loader(I + 1) = ZP + 2
1378             I += 1
1379         End If
1380     End If
1381 End If
1382 Next
1383
1384 ''ZP
1385     Instructions
1386     Types
1387     STA ZP
1388     ADC ZP
1389     STA (ZP),Y
1390     ADC ZP
1391     STA ZP
1392     STA (ZP),Y
1393     ADC ZP
1394     STA (ZP),Y
1395 ''ZP+1
1396     STA ZP+1
1397     DEC ZP+1
1398     LDA ZP+1
1399     DEC ZP+1
1400     LDA ZP+1
1401 ''Bits
1402     STA Bits
1403     ROR Bits
1404     STA Bits
1405     ASL Bits
1406     STA Bits
1407     ROL Bits
1408     STA Bits
1409
1410 Exit Sub
1411 Err:
1412     ErrCode = Err.Number
1413     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
1414
1415 End Sub
1416
1417 Public Function ConvertIntToHex(HInt As Integer, SLen As Integer) As String
1418     If DoOnError Then On Error GoTo Err
1419
1420     ConvertIntToHex = LCase(Hex(HInt))
1421
1422     If Len(ConvertIntToHex) < SLen Then
1423         ConvertIntToHex = Left(StrDup(SLen, "0"), SLen - ConvertIntToHex.Length) + ConvertIntToHex
1424     End If
1425
1426 Exit Function
1427 Err:
1428     ErrCode = Err.Number
1429     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
1430

```



```

1431 ConvertIntToHex = ""
1432
1433 End Function
1434
1435 Public Function UpdateBAM(DiskIndex As Integer) As Boolean
1436     If DoOnError Then On Error GoTo Err
1437
1438     UpdateBAM = True
1439
1440     Dim Cnt As Integer
1441     Dim B As Byte
1442     CP = Track(18)
1443
1444     For Cnt = &H90 To &HAA 'Name, ID, DOS type
1445         Disk(CP + Cnt) = &HA0
1446     Next
1447
1448     If DiskHeaderA(DiskIndex) = "" Then DiskHeaderA(DiskIndex) = "demo disk" + If(DiskCnt > 0, " "
+ (DiskIndex + 1).ToString, "")
1449
1450     '-----
1451
1452     For Cnt = 1 To Len(DiskHeaderA(DiskIndex))
1453         B = Ascii2PetSCII(Asc(Mid(DiskHeaderA(DiskIndex), Cnt, 1)))
1454         'If B > &H5F Then B -= &H20
1455         Disk(CP + &H8F + Cnt) = B
1456     Next
1457
1458     '-----
1459
1460     If DiskIDA(DiskIndex) <> "" Then
1461         For Cnt = 1 To Len(DiskIDA(DiskIndex))
1462             B = Ascii2PetSCII(Asc(Mid(DiskIDA(DiskIndex), Cnt, 1)))
1463             'If B > &H5F Then B -= &H20
1464             Disk(CP + &HA1 + Cnt) = B
1465         Next
1466     Else
1467         For Cnt = 1 To Len(DiskID)
1468             B = Ascii2PetSCII(Asc(Mid(DiskID, Cnt, 1)))
1469             'If B > &H5F Then B -= &H20
1470             Disk(CP + &HA1 + Cnt) = B
1471         Next
1472     End If
1473
1474     '-----
1475
1476     Exit Function
1477 Err:
1478     ErrCode = Err.Number
1479     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
1480
1481     UpdateBAM = False
1482
1483 End Function
1484
1485 Public Sub MakeTestDisk()
1486     If DoOnError Then On Error GoTo Err
1487
1488     Dim B As Byte
1489     Dim TDiff As Integer = Track(19) - Track(18)
1490     Dim SMax As Integer

```

```

1491 DemoStart = "0820"
1492 DemoName = "sparkle test"
1493 DiskHeader = "spakle test"
1494 DiskID = " 2019"
1495
1496
1497 NewDisk()
1498
1499 For T As Integer = 1 To 35
1500     Select Case T
1501         Case 1 To 17
1502             SMax = 20
1503         Case 19 To 24
1504             SMax = 18
1505         Case 25 To 30
1506             SMax = 17
1507         Case 31 To 35
1508             SMax = 16
1509     End Select
1510     If T <> 18 Then
1511         For S As Integer = 0 To SMax
1512             'For I As Integer = 0 To 255
1513             'Select Case (I And 15)
1514             'Case 0, 2, 4, 6, 9, 11, 13, 15
1515             'B = (I And 15) Xor &HF
1516             'Case Else
1517             'B = (I And 15) Xor &H6
1518             'End Select
1519             '
1520             'Select Case Int(I / 16)
1521             'Case 0, 2, 4, 6, 9, 11, 13, 15
1522             'B += (I And &HF0) Xor &HF0
1523             'Case Else
1524             'B += (I And &HF0) Xor &H60
1525             'End Select
1526             'Disk(Track(T) + S * 256 + I) = I
1527             'Next
1528             For I As Integer = S To S + 255
1529                 B = I Mod 256
1530                 Disk(Track(T) + S * 256 + I - S) = B
1531             Next
1532             DeleteBit(T, S, True)
1533         Next
1534     End If
1535 Next
1536
1537 InjectLoader(-1, 18, 7, 1, True)
1538 InjectDriveCode(1, 255, 1, True)
1539
1540 Exit Sub
1541 Err:
1542     ErrCode = Err.Number
1543     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
1544
1545 End Sub
1546
1547 Public Function BuildDemoFromScript(Optional SaveIt As Boolean = True) As Boolean
1548     If DoOnError Then On Error GoTo Err
1549
1550     TotLits = 0
1551     TotSM = 0

```

```
1552 TotNMM = 0
1553 TotFMM = 0
1554 TotNLM = 0
1555 TotFLM = 0
1556
1557 BuildDemoFromScript = True
1558
1559 'Generate Product ID unique to this build - it will be the same for all disks in this build
1560 Randomize()
1561 ProductID = Int(Rnd() * &HFFFFFF)
1562
1563 TotLit = 0 : TotMatch = 0
1564
1565 SS = 1 : SE = 1
1566
1567 'Check if this is a valid script
1568 If FindNextScriptEntry() = False Then GoTo NoDisk
1569
1570 If ScriptEntry <> ScriptHeader Then
1571     MsgBox("Invalid Loader Script file!", vbExclamation + vbOKOnly)
1572     GoTo NoDisk
1573 End If
1574
1575 TotalBits = 0
1576
1577 CurrentBundle = 0
1578 DiskCnt = -1
1579 TotalBundles = 0
1580 'DiskLoop = 0      'Reset Loop variable
1581 'Reset Disk Variables
1582 If ResetDiskVariables() = False Then GoTo NoDisk
1583 Dim NewD As Boolean = True
1584 NewBundle = False
1585 TmpSetNewBlock = False
1586
1587 FindNext:
1588     LastSS = SS
1589     LastSE = SE
1590     If FindNextScriptEntry() = False Then GoTo NoDisk
1591     'Split String
1592     If SplitScriptEntry() = False Then GoTo NoDisk
1593     'Set disk variables and add files
1594     Select Case LCase(ScriptEntryType)
1595     Case "path:"
1596         If NewD = False Then
1597             NewD = True
1598             If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1599             If ResetDiskVariables() = False Then GoTo NoDisk
1600         End If
1601         If ScriptEntryArray.Length > 0 Then
1602             D64Name = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
1603         End If
1604         NewBundle = True
1605     Case "header:"
1606         If NewD = False Then
1607             NewD = True
1608             If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1609             If ResetDiskVariables() = False Then GoTo NoDisk
1610         End If
1611         If ScriptEntryArray.Length > 0 Then
1612             DiskHeader = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
1613         End If
```

```

1614 NewBundle = True
1615 Case "id:"
1616     If NewD = False Then
1617         NewD = True
1618         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1619         If ResetDiskVariables() = False Then GoTo NoDisk
1620     End If
1621     If ScriptEntryArray.Length > 0 Then
1622         DiskID = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
1623     End If
1624     NewBundle = True
1625 Case "name:"
1626     If NewD = False Then
1627         NewD = True
1628         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1629         If ResetDiskVariables() = False Then GoTo NoDisk
1630     End If
1631     If ScriptEntryArray.Length > 0 Then
1632         DemoName = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
1633     End If
1634     NewBundle = True
1635 Case "start:"
1636     If NewD = False Then
1637         NewD = True
1638         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1639         If ResetDiskVariables() = False Then GoTo NoDisk
1640     End If
1641     If ScriptEntryArray.Length > 0 Then
1642         DemoStart = If(ScriptEntryArray(0) IsNot Nothing, ScriptEntryArray(0), "")
1643     End If
1644     NewBundle = True
1645 Case "dirart:"
1646     If NewD = False Then
1647         NewD = True
1648         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1649         If ResetDiskVariables() = False Then GoTo NoDisk
1650     End If
1651     If (ScriptEntryArray.Length > 0) AndAlso (ScriptEntryArray(0) <> "") Then
1652         If InStr(ScriptEntryArray(0), ":") = 0 Then
1653             ScriptEntryArray(0) = ScriptPath + ScriptEntryArray(0)
1654         End If
1655         If IO.File.Exists(ScriptEntryArray(0)) Then
1656             DirArtName = ScriptEntryArray(0)
1657             'DirArt = IO.File.ReadAllText(DirArtName)
1658         Else
1659             MsgBox("The following DirArt file does not exist:" + vbNewLine + vbNewLine +
ScriptEntryArray(0), vbOKOnly + vbExclamation, "DirArt file not found")
1660         End If
1661     End If
1662     NewBundle = True
1663 Case "zp:"
1664     If NewD = False Then
1665         NewD = True
1666         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1667         If ResetDiskVariables() = False Then GoTo NoDisk
1668     End If
1669     If DiskCnt = 0 Then
1670         If (ScriptEntryArray.Length > 0) AndAlso (ScriptEntryArray(0) IsNot Nothing) Then
1671             LoaderZP = ScriptEntryArray(0) 'ZP usage can only be set from first disk
1672         End If
1673     End If
1674     NewBundle = True

```

```

1675 Case "packer:"
1676     If NewD = False Then
1677         NewD = True
1678         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1679         If ResetDiskVariables() = False Then GoTo NoDisk
1680     End If
1681     If ScriptEntryArray.Length > 0 Then
1682         If ScriptEntryArray(0) IsNot Nothing Then
1683             Packer = If(LCase(ScriptEntryArray(0)) = "better", PackerTypes.Better,
PackerTypes.Faster)
1684         End If
1685     End If
1686     NewBundle = True
1687 Case "il0:"
1688     If NewD = False Then
1689         NewD = True
1690         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1691         If ResetDiskVariables() = False Then GoTo NoDisk
1692     End If
1693     Dim TmpIL As Integer = If(ScriptEntryArray.Length > 0,
Convert.ToInt32(ScriptEntryArray(0), 10), 0)
1694     IL0 = If(TmpIL Mod 21 > 0, TmpIL Mod 21, DefaultIL0)
1695     NewBundle = True
1696 Case "il1:"
1697     If NewD = False Then
1698         NewD = True
1699         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1700         If ResetDiskVariables() = False Then GoTo NoDisk
1701     End If
1702     Dim TmpIL As Integer = If(ScriptEntryArray.Length > 0,
Convert.ToInt32(ScriptEntryArray(0), 10), 0)
1703     IL1 = If(TmpIL Mod 19 > 0, TmpIL Mod 19, DefaultIL1)
1704     NewBundle = True
1705 Case "il2:"
1706     If NewD = False Then
1707         NewD = True
1708         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1709         If ResetDiskVariables() = False Then GoTo NoDisk
1710     End If
1711     Dim TmpIL As Integer = If(ScriptEntryArray.Length > 0,
Convert.ToInt32(ScriptEntryArray(0), 10), 0)
1712     IL2 = If(TmpIL Mod 18 > 0, TmpIL Mod 18, DefaultIL2)
1713     NewBundle = True
1714 Case "il3:"
1715     If NewD = False Then
1716         NewD = True
1717         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1718         If ResetDiskVariables() = False Then GoTo NoDisk
1719     End If
1720     Dim TmpIL As Integer = If(ScriptEntryArray.Length > 0,
Convert.ToInt32(ScriptEntryArray(0), 10), 0)
1721     IL3 = If(TmpIL Mod 17 > 0, TmpIL Mod 17, DefaultIL3)
1722     NewBundle = True
1723 Case "prodid:"
1724     If NewD = False Then
1725         NewD = True
1726         If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1727         If ResetDiskVariables() = False Then GoTo NoDisk
1728     End If
1729     If ScriptEntryArray.Length > 0 Then
1730         If IsNumeric("&H" + ScriptEntryArray(0)) Then
1731             ProductID = Convert.ToInt32(ScriptEntryArray(0), 16)
1732         Else

```

```

1733 vbNewLine + vbNewLine +
1734 "Sparkle will generate a pseudorandom Product ID.", vbOKOnly +
vbExclamation, "Product ID Error")
1735 End If
1736 End If
1737 NewBundle = True
1738 Case "tracks:"
1739 If NewD = False Then
1740 NewD = True
1741 If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1742 If ResetDiskVariables() = False Then GoTo NoDisk
1743 End If
1744 If ScriptEntryArray.Length > 0 Then
1745 If IsNumeric(ScriptEntryArray(0)) Then
1746 Dim TmpTracks As Integer = Convert.ToInt32(ScriptEntryArray(0), 10)
1747 If TmpTracks = ExtTracksPerDisk Then
1748 TracksPerDisk = ExtTracksPerDisk
1749 SectorsPerDisk = ExtSectorsPerDisk
1750 Else
1751 TracksPerDisk = StdTracksPerDisk
1752 SectorsPerDisk = StdSectorsPerDisk
1753 End If
1754 UpdateDiskSizeOnTheFly()
1755 End If
1756 End If
1757 NewBundle = True
1758 Case "hsfile:"
1759 If NewD = False Then
1760 NewD = True
1761 If FinishDisk(False, SaveIt) = False Then GoTo NoDisk
1762 If ResetDiskVariables() = False Then GoTo NoDisk
1763 End If
1764 If ScriptEntryArray(0) <> "" Then
1765 If AddHSFile() = False Then GoTo NoDisk
1766 End If
1767 NewBundle = True
1768 Case "script:"
1769 If InsertScript(ScriptEntryArray(0)) = False Then GoTo NoDisk
1770 NewBundle = True 'Files in the embedded script will ALWAYS be in a new bundle (i.e.
scripts cannot be embedded in a bundle)!!!
1771 Case "file:"
1772 'Add files to bundle array, if new bundle=true, we will first sort, compress and add
previous bundle to disk
1773 If AddFile() = False Then GoTo NoDisk
1774 NewD = False 'We have added at least one file to this disk, so next disk info entry
will be a new disk
1775 NewBundle = False
1776 Case "mem:"
1777 If AddVirtualFile() = False Then GoTo NoDisk
1778 NewBundle = False
1779 'NewD = False 'IS THIS NEEDED???'
1780 'Case "sort:"
1781 Case "align"
1782 If NewD = False Then
1783 TmpSetNewBlock = True
1784 End If
1785 Case Else
1786 If NewBundle = True Then
1787 If BundleDone() = False Then GoTo NoDisk
1788 NewBundle = False
1789 End If
1790 End Select

```

```

1791         If SE < Script.Length Then GoTo FindNext
1792
1793
1794         'Last disk: sort, compress and add last bundle, then update bundle count, add loader & drive
code, save disk, and we are done :)
1795         If FinishDisk(True, SaveIt) = False Then GoTo NoDisk
1796
1797         'MsgBox(TotalBits.ToString)
1798         'MsgBox("Literals:" + vbTab + vbTab + vbTab + TotLits.ToString + vbNewLine +
1799         '"Short Matches:" + vbTab + vbTab + TotSM.ToString + vbNewLine +
1800         '"Near Mid Matches:" + vbTab + TotNMM.ToString + vbNewLine +
1801         '"Near Long Matches:" + vbTab + TotNLM.ToString + vbNewLine +
1802         '"Far Mid Matches:" + vbTab + vbTab + TotFMM.ToString + vbNewLine +
1803         '"Far Long Matches:" + vbTab + vbTab + TotFLM.ToString)
1804
1805         Exit Function
1806 Err:
1807         ErrCode = Err.Number
1808         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
1809 NoDisk:
1810         BuildDemoFromScript = False
1811         If ErrCode = 0 Then ErrCode = -1
1812
1813     End Function
1814
1815     Public Function InsertScript(SubScriptPath As String) As Boolean
1816         If DoOnError Then On Error GoTo Err
1817
1818         InsertScript = True
1819
1820         Dim SPath As String = SubScriptPath
1821
1822         'Calculate full path
1823         If InStr(SubScriptPath, ":") = 0 Then SubScriptPath = ScriptPath + SubScriptPath
1824
1825         If IO.File.Exists(SubScriptPath) = False Then
1826             MsgBox("The following script was not found and could not be processed:" + vbNewLine +
vbNewLine + SubScriptPath, vbOKOnly + vbExclamation, "Script not found")
1827             InsertScript = False
1828             Exit Function
1829         End If
1830
1831         'Find relative path of subscript
1832         For I As Integer = Len(SPath) - 1 To 0 Step -1
1833             If Right(SPath, 1) <> "\" Then
1834                 SPath = Left(SPath, Len(SPath) - 1)
1835             Else
1836                 Exit For
1837             End If
1838         Next
1839
1840         'Find relative path of subscript - THIS DOESN'T WORK IF THERE IS NO "\" IN THE SUBSCRIPTS PATH
1841         'WOULD PROBABLY WORK WITH THE ADDITIONAL INSTR() CHECK BUT ANYWAY, LET'S USE THE OLD AND
PROVEN CODE ABOVE...
1842         'If InStr(SPath, "\") <> 0 Then
1843         'For I As Integer = Len(SPath) To 1 Step -1
1844         'If Mid(SPath, I, 1) = "\" Then
1845         'SPath = Strings.Left(SPath, I)           'Path
1846         'Exit For
1847         'End If
1848         'Next
1849         'Else

```

```

1850 'SPath = ""
1851 'End If
1852
1853 Dim Lines() As String = Split(IO.File.ReadAllText(SubScriptPath), vbCrLf)
1854
1855 Dim S As String = ""
1856 For I As Integer = 0 To Lines.Count - 1
1857     Lines(I) = Lines(I).TrimEnd(Chr(13)) 'Trim vbCR from end of lines if vbCrLf was used
1858
1859     If InStr(Lines(I), vbTab) = 0 Then
1860         ScriptEntryType = Replace(Lines(I), "", "")
1861         ScriptEntry = ""
1862     Else
1863         ScriptEntryType = Replace(Strings.Left(Lines(I), InStr(Lines(I), vbTab) - 1), " ", "")
1864         ScriptEntry = Strings.Right(Lines(I), Len(Lines(I)) - InStr(Lines(I),
vbTab)).TrimStart(vbTab)
1865     End If
1866
1867     SplitEntry()
1868
1869     'Skip Script Header
1870     If Lines(I) <> ScriptHeader Then
1871         If S <> "" Then
1872             S += vbCrLf
1873         End If
1874         'Add relative path of subscript to relative path of subscript entries
1875         Select Case LCase(ScriptEntryType)
1876             Case "file:"
1877                 If ScriptEntryArray(0) IsNot Nothing Then
1878                     If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath +
ScriptEntryArray(0)
1879                 End If
1880                 Lines(I) = "File:" + vbTab + ScriptEntryArray(0)
1881                 For J As Integer = 1 To ScriptEntryArray.Length - 1
1882                     If ScriptEntryArray(J) IsNot Nothing Then
1883                         Lines(I) += vbTab + ScriptEntryArray(J)
1884                     End If
1885                 Next
1886             Case "mem:"
1887                 If ScriptEntryArray(0) IsNot Nothing Then
1888                     If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath +
ScriptEntryArray(0)
1889                 End If
1890                 Lines(I) = "Mem:" + vbTab + ScriptEntryArray(0)
1891                 For J As Integer = 1 To ScriptEntryArray.Length - 1
1892                     If ScriptEntryArray(J) IsNot Nothing Then
1893                         Lines(I) += vbTab + ScriptEntryArray(J)
1894                     End If
1895                 Next
1896             Case "script:"
1897                 If ScriptEntryArray(0) IsNot Nothing Then
1898                     If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath +
ScriptEntryArray(0)
1899                 End If
1900                 Lines(I) = "Script:" + vbTab + ScriptEntryArray(0)
1901             Case "path:"
1902                 If ScriptEntryArray(0) IsNot Nothing Then
1903                     If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath +
ScriptEntryArray(0)
1904                 End If
1905                 Lines(I) = "Path:" + vbTab + ScriptEntryArray(0)
1906             Case "dirart:"
1907                 If ScriptEntryArray(0) IsNot Nothing Then

```



```

1908 If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath +
ScriptEntryArray(0)
1909 End If
1910 Lines(I) = "DirArt:" + vbTab + ScriptEntryArray(0)
1911 Case "hsfile:"
1912 If ScriptEntryArray(0) IsNot Nothing Then
1913 If InStr(ScriptEntryArray(0), ":") = 0 Then ScriptEntryArray(0) = SPath +
ScriptEntryArray(0)
1914 End If
1915 Lines(I) = "HSFile:" + vbTab + ScriptEntryArray(0)
1916 For J As Integer = 1 To ScriptEntryArray.Length - 1
1917 If ScriptEntryArray(J) IsNot Nothing Then
1918 Lines(I) += vbTab + ScriptEntryArray(J)
1919 End If
1920 Next
1921 End Select
1922 'If Strings.Right(Lines(I), 1) <> vbLf Then
1923 'Lines(I) += vbLf
1924 'End If
1925
1926
1927 'If Left(LCase(Lines(I)), 5) = "file:" Then
1928 'If (InStr(Right(Lines(I), Len(Lines(I)) - 5), ":") = 0) And (InStr(Right(Lines(I),
Len(Lines(I)) - 5), SPath) = 0) Then
1929 'Lines(I) = "File:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) -
5).TrimStart(vbTab) 'Trim any extra leading TABs
1930 'End If
1931 'ElseIf Left(LCase(Lines(I)), 7) = "script:" Then
1932 'If (InStr(Right(Lines(I), Len(Lines(I)) - 7), ":") = 0) And (InStr(Right(Lines(I),
Len(Lines(I)) - 7), SPath) = 0) Then
1933 'Lines(I) = "Script:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) -
7).TrimStart(vbTab)
1934 'End If
1935 'ElseIf Left(LCase(Lines(I)), 5) = "list:" Then
1936 'If (InStr(Right(Lines(I), Len(Lines(I)) - 5), ":") = 0) And (InStr(Right(Lines(I),
Len(Lines(I)) - 5), SPath) = 0) Then
1937 'Lines(I) = "Script:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) -
5).TrimStart(vbTab)
1938 'End If
1939 'ElseIf Left(LCase(Lines(I)), 5) = "path:" Then
1940 'If (InStr(Right(Lines(I), Len(Lines(I)) - 5), ":") = 0) And (InStr(Right(Lines(I),
Len(Lines(I)) - 5), SPath) = 0) Then
1941 'Lines(I) = "Path:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) -
5).TrimStart(vbTab)
1942 'End If
1943 'ElseIf Left(LCase(Lines(I)), 7) = "dirart:" Then
1944 'If (InStr(Right(Lines(I), Len(Lines(I)) - 7), ":") = 0) And (InStr(Right(Lines(I),
Len(Lines(I)) - 7), SPath) = 0) Then
1945 'Lines(I) = "DirArt:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) -
7).TrimStart(vbTab)
1946 'End If
1947 'ElseIf Left(LCase(Lines(I)), 7) = "hsfile:" Then
1948 'If (InStr(Right(Lines(I), Len(Lines(I)) - 7), ":") = 0) And (InStr(Right(Lines(I),
Len(Lines(I)) - 7), SPath) = 0) Then
1949 'Lines(I) = "HSFile:" + vbTab + SPath + Right(Lines(I), Len(Lines(I)) -
7).TrimStart(vbTab)
1950 'End If
1951 'End If
1952 S += Lines(I)
1953 End If
1954 Next
1955
1956 Script = Replace(Script, ScriptLine, S)
1957

```

```

1958         SS = LastSS
1959         SE = LastSE
1960
1961         Exit Function
1962 Err:
1963         ErrCode = Err.Number
1964         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
1965
1966         InsertScript = False
1967
1968     End Function
1969
1970 Private Function AddHeaderAndID() As Boolean
1971     If DoOnError Then On Error GoTo Err
1972
1973     AddHeaderAndID = True
1974
1975     Dim B As Byte
1976
1977     CP = Track(18)
1978
1979     For Cnt As Integer = &H90 To &HAA
1980         Disk(CP + Cnt) = &HA0
1981     Next
1982
1983     If Len(DiskHeader) > 16 Then
1984         DiskHeader = Left(DiskHeader, 16)
1985     End If
1986
1987     If DiskHeader = "" Then
1988         For Cnt As Integer = 1 To 16
1989             Disk(CP + &H8F + Cnt) = 32
1990         Next
1991     Else
1992         For Cnt As Integer = 1 To Len(DiskHeader)
1993             B = Ascii2PetSCII(Asc(Mid(DiskHeader, Cnt, 1)))
1994             'If B > &H5F Then B -= &H20
1995             Disk(CP + &H8F + Cnt) = B
1996         Next
1997     End If
1998
1999     If Len(DiskID) > 5 Then
2000         DiskID = Left(DiskID, 5)
2001     End If
2002
2003     If DiskID = "" Then
2004         For Cnt As Integer = 1 To 5                                     'Overwrites Disk ID and DOS type (5
characters max.)
2005             Disk(CP + &HA1 + Cnt) = 32
2006         Next
2007     Else
2008         For Cnt As Integer = 1 To Len(DiskID)                           'Overwrites Disk ID and DOS type (5
characters max.)
2009             B = Ascii2PetSCII(Asc(Mid(DiskID, Cnt, 1)))
2010             'If B > &H5F Then B -= &H20
2011             Disk(CP + &HA1 + Cnt) = B
2012         Next
2013     End If
2014
2015
2016     Exit Function

```

```

2017 Err:
2018     ErrCode = Err.Number
2019     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
2020
2021     AddHeaderAndID = False
2022
2023 End Function
2024
2025 Private Function FinishDisk(LastDisk As Boolean, Optional SaveIt As Boolean = True) As Boolean
2026     If DoOnErr Then On Error GoTo Err
2027
2028     FinishDisk = True
2029
2030     If (BundleCnt = 0) And (FileCnt = -1) Then
2031 disk")         MsgBox("This disk does not contain any files!", vbOKOnly + vbExclamation, "Unable to build
2032                 GoTo NoDisk
2033     End If
2034     If BundleDone() = False Then GoTo NoDisk
2035     If CompressBundle() = False Then GoTo NoDisk
2036     If CloseBundle(0, True) = False Then GoTo NoDisk
2037     If CloseBuffer() = False Then GoTo NoDisk
2038
2039     If MaxBundleNoExceeded Then
2040 vbNewLine +     MsgBox("The number of file bundles is greater than 128 on this disk!" + vbNewLine +
2041 be loaded using the LoadNext function.", vbOKOnly + vbInformation, "More than 128 bundles on disk")
2042     End If
2043
2044     'Now add compressed parts to disk
2045     If AddCompressedBundlesToDisk() = False Then GoTo NoDisk
2046     If InjectLoader(-1, 18, 7, 1) = False Then GoTo NoDisk
2047     If InjectDriveCode(DiskCnt, LoaderBundles, If(LastDisk = False, DiskCnt + 1, &H80)) = False
Then GoTo NoDisk
2048
2049     AddHeaderAndID()
2050     AddDemoNameToDisk(-1, 18, 7)
2051     AddDirArt()
2052
2053     BytesSaved += Int(BitsSaved / 8)
2054     BitsSaved = BitsSaved Mod 8
2055
2056     UpdateBlocksFree()
2057
2058     If SaveIt = True Then
2059         If SaveDisk() = False Then GoTo NoDisk
2060     End If
2061
2062     Exit Function
2063 Err:
2064     ErrCode = Err.Number
2065     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
2066 NoDisk:
2067     FinishDisk = False
2068
2069 End Function
2070
2071 Private Function SaveDisk() As Boolean
2072     'CANNOT HAVE On Error FUNCTION DUE TO TRY/CATCH
2073

```

```

2074 SaveDisk = True
2075
2076 If D64Name = "" Then D64Name = "Demo Disk " + (DiskCnt + 1).ToString + ".d64"
2077
2078 If InStr(D64Name, ":") = 0 Then
2079     D64Name = ScriptPath + D64Name
2080 End If
2081
2082 Dim SaveCtr As Integer = 20
2083
2084 TryAgain:
2085     ErrCode = 0
2086     Try
2087         If CmdLine = True Then
2088             'We are in command line, just save the disk
2089             IO.File.WriteAllBytes(D64Name, Disk)
2090         Else
2091             'We are in app mode, show dialog
2092             Dim SaveDLG As New SaveFileDialog With {
2093                 .Filter = "D64 Files (*.d64)|*.d64",
2094                 .Title = "Save D64 File As...",
2095                 .FileName = D64Name,
2096                 .RestoreDirectory = True
2097             }
2098
2099             Dim R As DialogResult = SaveDLG.ShowDialog(FrmMain)
2100
2101             If R = DialogResult.OK Then
2102                 D64Name = SaveDLG.FileName
2103                 If Right(D64Name, 4) <> ".d64" Then
2104                     D64Name += ".d64"
2105                 End If
2106                 IO.File.WriteAllBytes(D64Name, Disk)
2107                 FileChanged = False
2108             Else
2109                 FileChanged = True
2110             End If
2111
2112         End If
2113     Catch ex As Exception
2114         If CmdLine = True Then
2115             If SaveCtr > 0 Then
2116                 Threading.Thread.Sleep(20) 'If file could not be saved, wait 20 msec and try
again 20 times before showing error message
2117                 SaveCtr -= 1
2118                 GoTo TryAgain
2119             End If
2120         End If
2121         ErrCode = Err.Number 'Save error code here
2122         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
2123             Err.Clear()
2124             SaveCtr = 20
2125             GoTo TryAgain
2126         Else
2127             SaveDisk = False
2128             FileChanged = True
2129         End If
2130     End Try
2131
2132 End Function

```

```

2133     Public Function CompressBundle(Optional FromEditor = False) As Boolean
2134         If DoOnErr Then On Error GoTo Err
2135
2136
2137         CompressBundleFromEditor = FromEditor
2138
2139         CompressBundle = True
2140
2141         Dim PreBCnt As Integer = BufferCnt
2142
2143         If Prgs.Count = 0 Then Exit Function           'GoTo NoComp DOES NOT WORK!!!
2144
2145         'DO NOT RESET ByteSt AND BUFFER VARIABLES HERE!!!
2146
2147         If (BufferCnt = 0) And (BytePtr = 255) Then
2148             NewBlock = SetNewBlock           'SetNewBlock is true at closing the previous bundle, so
first it just sets NewBlock2
2149             SetNewBlock = False             'And NewBlock will fire at the desired bundle
2150         Else
2151             If FromEditor = False Then       'Don't finish previous bundle here if we are calculating
bundle size from Editor
2152
2153                 '-----
2154                 '"SPRITE BUG"
2155                 'Compression bug involving the transitional block - FIXED
2156                 'Fix: include the I/O status of the first file of this bundle in the calculation for
2157                 'finishing the previous bundle
2158                 '-----
2159
2160                 'Before finishing the previous bundle, calculate I/O status of the LAST BYTE of the
first file of this bundle
2161                 '(Files already sorted)
2162                 Dim ThisBundleIO As Integer = If(FileIOA.Count > 0, CheckNextIO(FileAddrA(0),
FileLenA(0), FileIOA(0)), 0)
2163                 If CloseBundle(ThisBundleIO, False) = False Then GoTo NoComp
2164             End If
2165         End If
2166
2167         '-----
2168         'SAVE CURRENT BIT POINTER AND BUFFER COUNT FOR DIRECTORY
2169         '-----
2170
2171         If FromEditor = False Then
2172             'Only if we are NOT in the Editor
2173             If BundleNo < 128 Then
2174                 DirBlocks((BundleNo * 4) + 3) = BitPtr
2175                 DirPtr(BundleNo) = BufferCnt
2176                 BundleNo += 1
2177             Else
2178                 MaxBundleNoExceeded = True
2179             End If
2180         End If
2181
2182         '-----
2183
2184         NewBundle = True
2185         LastFileOfBundle = False
2186
2187         PartialFileIndex = -1
2188
2189         For I As Integer = 0 To Prgs.Count - 1
2190             'Mark the last file in a bundle for better compression
2191             If I = Prgs.Count - 1 Then LastFileOfBundle = True

```

```

2192 'The only two parameters that are needed are FA and FUIO... FileLenA(i) is not used
2193
2194 If PartialFileIndex = -1 Then PartialFileOffset = Prgs(I).ToArray.Length - 1
2195
2196 PackFile(Prgs(I).ToArray, I, FileAddrA(I), FileIOA(I))
2197 If I < Prgs.Count - 1 Then
2198     'WE NEED TO USE THE NEXT FILE'S ADDRESS, LENGTH AND I/O STATUS HERE
2199     'FOR I/O BYTE CALCULATION FOR THE NEXT PART - BUG reported by Raistlin/G*P
2200     PrgAdd = Convert.ToInt32(FileAddrA(I + 1), 16)
2201     PrgLen = Prgs(I + 1).Length ' Convert.ToInt32(FileLenA(I + 1), 16)
2202     FileUnderIO = FileIOA(I + 1)
2203     CloseFile()
2204 End If
2205 Next
2206
2207 LastBlockCnt = BlockCnt
2208
2209 If LastBlockCnt > 255 Then
2210     'Parts cannot be larger than 255 blocks compressed
2211     'There is some confusion here how PartCnt is used in the Editor and during Disk
building...
2212     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!")
2213     If CompressBundleFromEditor = False Then GoTo NoComp
2214 End If
2215
2216 'IF THE WHOLE Bundle IS LESS THAN 1 BLOCK, THEN "IT DOES NOT COUNT", Bundle Counter WILL NOT
BE INCREASED
2217 If PreBCnt = BufferCnt Then
2218     BundleCnt -= 1
2219 End If
2220
2221 Exit Function
2222 Err:
2223     ErrCode = Err.Number
2224     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
2225 NoComp:
2226     CompressBundle = False
2227
2228 End Function
2229
2230 Private Function AddFile() As Boolean
2231     If DoOnError Then On Error GoTo Err
2232
2233     AddFile = True
2234
2235     If NewBundle = True Then
2236         If BundleDone() = False Then GoTo NoDisk
2237     End If
2238
2239     'Then add file to bundle
2240     If AddFileToBundle() = False Then GoTo NoDisk
2241
2242     Exit Function
2243 Err:
2244     ErrCode = Err.Number
2245     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
2246 NoDisk:
2247     AddFile = False

```

```

2248 End Function
2249 Private Function BundleDone() As Boolean
2250     If DoOnError Then On Error GoTo Err
2251
2252     BundleDone = True
2253
2254     'First finish last bundle, if it exists
2255     If tmpPrgs.Count > 0 Then
2256         CurrentBundle += 1
2257
2258         'Sort files in bundle
2259         If SortBundle() = False Then GoTo NoDisk
2260         '-----
2261         'Then compress files and add them to bundle
2262         If CompressBundle() = False Then GoTo NoDisk           'THIS WILL RESET NewPart TO FALSE
2263
2264         Prgs = tmpPrgs.ToList
2265         FileNameA = tmpFileNameA
2266         FileAddrA = tmpFileAddrA
2267         FileOffsA = tmpFileOffsA
2268         FileLenA = tmpFileLenA
2269         FileIOA = tmpFileIOA
2270         SetNewBlock = TmpSetNewBlock
2271         TmpSetNewBlock = False
2272
2273         VFiles = tmpVFiles.ToList
2274         VFileNameA = tmpVFileNameA
2275         VFileAddrA = tmpVFileAddrA
2276         VFileOffsA = tmpVFileOffsA
2277         VFileLenA = tmpVFileLenA
2278         VFileIOA = tmpVFileIOA
2279
2280         '-----
2281         'Then reset bundle variables (file arrays, prg array, block cnt), increase bundle counter
2282         ResetBundleVariables()
2283     End If
2284
2285     Exit Function
2286
2287 Err:
2288     ErrCode = Err.Number
2289     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
2290 + " Error")
2291 NoDisk:
2292     BundleDone = False
2293
2294 End Function
2295
2296 Public Function CheckNextIO(sAddress As String, sLength As String, NextFileUnderIO As Boolean) As
Integer
2297     If DoOnError Then On Error GoTo Err
2298
2299     Dim pAddress As Integer = Convert.ToInt32(sAddress, 16) + Convert.ToInt32(sLength, 16)
2300
2301     If pAddress < 256 Then           'Are we loading to the Zero Page? If yes, we need to signal it by
adding IO Flag
2302         CheckNextIO = 1
2303     Else
2304         CheckNextIO = If((pAddress >= &HD000) And (pAddress <= &HFFFF) And (NextFileUnderIO =
True), 1, 0)
2305     End If
2306

```

```

2307 Exit Function
2308 Err:
2309     ErrCode = Err.Number
2310     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
2311
2312 End Function
2313
2314 Public Function SortBundle() As Boolean
2315     If DoOnErr Then On Error GoTo Err
2316
2317     SortBundle = True
2318
2319     If tmpPrgs.Count = 0 Then Exit Function
2320     If tmpPrgs.Count = 1 Then GoTo SortDone
2321
2322     Dim Change As Boolean
2323     Dim FSO, FEO, FSI, FEI As Integer 'File Start and File End Outer loop/Inner loop
2324     Dim PO(), PI() As Byte
2325     Dim S As String
2326     Dim bIO As Boolean
2327
2328     '-----
2329     'Check files for overlap
2330
2331     For O As Integer = 0 To tmpPrgs.Count - 2
2332         FSO = Convert.ToInt32(tmpFileAddrA(O), 16) 'Outer loop File Start
2333         FEO = FSO + Convert.ToInt32(tmpFileLenA(O), 16) - 1 'Outer loop File End
2334         For I As Integer = O + 1 To tmpPrgs.Count - 1
2335             FSI = Convert.ToInt32(tmpFileAddrA(I), 16) 'Inner loop File Start
2336             FEI = FSI + Convert.ToInt32(tmpFileLenA(I), 16) - 1 'Inner loop File End
2337             '--|-----+-----|----OR----|-----+-----|----OR----|-----+-----|----OR----|-----
+-----|--
2338             ' FSO FSI FEO FSO FEI FEO FSI FSO FEI FSI
2339             FEI FEI
2340             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
2341                 Dim OLS As Integer = If(FSO >= FSI, FSO, FSI) 'Overlap Start address
2342                 Dim OLE As Integer = If(FEO <= FEI, FEO, FEI) 'Overlap End address
2343
2344                 If (OLS >= &HD000) And (OLE <= &HFFFF) And (tmpFileIOA(O) <> tmpFileIOA(I)) Then
2345                     'Overlap is IO memory only and different IO status - NO OVERLAP
2346                     Else
2347                         MsgBox("The following two files overlap in Bundle " + (BundleCnt - 1).ToString
+ ":" _
2348                         + vbNewLine + vbNewLine + tmpFileNameA(I) + " ($" + Hex(FSI) + " - $" +
Hex(FEI) + ")" + vbNewLine + vbNewLine _
2349                         + tmpFileNameA(O) + " ($" + Hex(FSO) + " - $" + Hex(FEO) + ")", vbOKOnly +
vbExclamation)
2350                     End If
2351                 End If
2352             Next
2353         Next
2354     '-----
2355     'Append adjacent files
2356 Restart:
2357     Change = False
2358
2359     For O As Integer = 0 To tmpPrgs.Count - 2
2360         FSO = Convert.ToInt32(tmpFileAddrA(O), 16)
2361         FEO = Convert.ToInt32(tmpFileLenA(O), 16)
2362         For I As Integer = O + 1 To tmpPrgs.Count - 1

```



```

2363 FSI = Convert.ToInt32(tmpFileAddrA(I), 16)
2364 FEI = Convert.ToInt32(tmpFileLenA(I), 16)
2365
2366 If FSO + FEO = FSI Then
2367     'Inner file follows outer file immediately
2368     If (FSI <= &HD000) Or (FSI > &HDEFF) Then
2369         'Append files as they meet outside IO memory
2370 Append: PO = tmpPrgs(0)
2371         PI = tmpPrgs(I)
2372         ReDim Preserve PO(FEO + FEI - 1)
2373
2374         For J As Integer = 0 To FEI - 1
2375             PO(FEO + J) = PI(J)
2376         Next
2377
2378         tmpPrgs(0) = PO
2379
2380         Change = True
2381     Else
2382         If tmpFileIOA(0) = tmpFileIOA(I) Then
2383             'Files meet inside IO memory, append only if their IO status is the same
2384             GoTo Append
2385         End If
2386     End If
2387 ElseIf FSI + FEI = FSO Then
2388     'Outer file follows inner file immediately
2389     If (FSO <= &HD000) Or (FSO > &HDEFF) Then
2390         'Prepend files as they meet outside IO memory
2391 Prepend: PO = tmpPrgs(0)
2392         PI = tmpPrgs(I)
2393         ReDim Preserve PI(FEI + FEO - 1)
2394
2395         For J As Integer = 0 To FEO - 1
2396             PI(FEI + J) = PO(J)
2397         Next
2398
2399         tmpPrgs(0) = PI
2400
2401         tmpFileAddrA(0) = tmpFileAddrA(I)
2402
2403         Change = True
2404     Else
2405         If tmpFileIOA(0) = tmpFileIOA(I) Then
2406             'Files meet inside IO memory, prepend only if their IO status is the same
2407             GoTo Prepend
2408         End If
2409     End If
2410 End If
2411
2412 If Change = True Then
2413     'Update merged file's IO status
2414 RAISTLIN/G*P tmpFileIOA(0) = tmpFileIOA(0) Or tmpFileIOA(I) 'BUG FIX - REPORTED BY
2415
2416     'New file's length is the length of the two merged files
2417     FEO += FEI
2418
2419     tmpFileLenA(0) = ConvertIntToHex(FEO, 4)
2420     'Remove File(I) and all its parameters
2421     For J As Integer = I To tmpPrgs.Count - 2
2422         tmpFileNameA(J) = tmpFileNameA(J + 1)
2423         tmpFileAddrA(J) = tmpFileAddrA(J + 1)
2424         tmpFileOffsA(J) = tmpFileOffsA(J + 1) 'this may not be needed later

```

```

2424         tmpFileLenA(J) = tmpFileLenA(J + 1)
2425         tmpFileIOA(J) = tmpFileIOA(J + 1)
2426     Next
2427     'One less file left
2428     FileCnt -= 1
2429     ReDim Preserve tmpFileNameA(tmpPrgs.Count - 2), tmpFileAddrA(tmpPrgs.Count - 2),
tmpFileOffsA(tmpPrgs.Count - 2), tmpFileLenA(tmpPrgs.Count - 2)
2430     ReDim Preserve tmpFileIOA(tmpPrgs.Count - 2)
2431     tmpPrgs.Remove(tmpPrgs(I))
2432     GoTo Restart
2433 End If
2434 Next
2435 Next
2436
2437 '-----
2438 'Sort files by length (short files first, thus, last block will more likely contain 1 file
only = faster depacking)
2439 ReSort:
2440     Change = False
2441     For I As Integer = 0 To tmpPrgs.Count - 2
2442         'Sort except if file length < 4, to allow for ZP relocation script hack
2443         If Convert.ToInt32(tmpFileAddrA(I), 16) < Convert.ToInt32(tmpFileAddrA(I + 1), 16) Then
2444             PI = tmpPrgs(I)
2445             tmpPrgs(I) = tmpPrgs(I + 1)
2446             tmpPrgs(I + 1) = PI
2447
2448             S = tmpFileNameA(I)
2449             tmpFileNameA(I) = tmpFileNameA(I + 1)
2450             tmpFileNameA(I + 1) = S
2451
2452             S = tmpFileAddrA(I)
2453             tmpFileAddrA(I) = tmpFileAddrA(I + 1)
2454             tmpFileAddrA(I + 1) = S
2455
2456             S = tmpFileOffsA(I)
2457             tmpFileOffsA(I) = tmpFileOffsA(I + 1)
2458             tmpFileOffsA(I + 1) = S
2459
2460             S = tmpFileLenA(I)
2461             tmpFileLenA(I) = tmpFileLenA(I + 1)
2462             tmpFileLenA(I + 1) = S
2463
2464             bIO = tmpFileIOA(I)
2465             tmpFileIOA(I) = tmpFileIOA(I + 1)
2466             tmpFileIOA(I + 1) = bIO
2467             Change = True
2468         End If
2469     Next
2470     If Change = True Then GoTo ReSort
2471
2472 SortDone:
2473     '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
2474     'to finish the last block of the previous bundle (when the I/O status of the just sorted
bundle needs to be known)
2475     'This is used in CloseBuffer
2476
2477     '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
2478     '+/- 1 Match Bit (if the last sequence of the last bundle is a match sequence, no Match Bit
after a Literal sequence)
2479     'Match Bit will be determined by MLen in SequenceFits() function, NOT ADDED TO
BitsNeededForNextBundle here!!!

```

```

2480      'We may be overcalculating here but that is safer than undercalculating which would result in
2481      buggy decompression
2482      'If the last block is not the actual last block of the bundle...
2483      'With overcalculation, worst case scenario is a little bit worse compression ratio of the last
2484      block
2485      BitsNeededForNextBundle = (6 + CheckNextIO(tmpFileAddrA(0), tmpFileLenA(0), tmpFileIOA(0))) *
2486      8
2487      ' +/- 1 Match Bit which will be added later in CloseBuffer if needed
2488      Exit Function
2489      Err:
2490      ErrCode = Err.Number
2491      MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
2492      + " Error")
2493      NoSort:
2494      SortBundle = False
2495
2496      End Function
2497
2498      Public Function AddHSFile() As Boolean
2499      If DoOnError Then On Error GoTo Err
2500
2501      AddHSFile = True
2502
2503      Dim FN As String = ScriptEntryArray(0)
2504      Dim FA As String = ""
2505      Dim FO As String = ""
2506      Dim FL As String = ""
2507      Dim FAN As Integer = 0
2508      Dim FON As Integer = 0
2509      Dim FLN As Integer = 0
2510
2511      Dim NumParams As Integer = 1
2512
2513      Dim P() As Byte
2514
2515      If InStr(FN, ":") = 0 Then          'relative file path
2516      FN = ScriptPath + FN              'look for file in script's folder
2517      End If
2518
2519      'Correct file parameter length to 4-8 characters
2520      For I As Integer = 1 To ScriptEntryArray.Count - 1
2521      If ParameterIsNumeric(I) Then
2522      NumParams += 1
2523      Else
2524      Exit For
2525      End If
2526
2527      CorrectParameterStringLength(I)
2528
2529      ''Remove HEX prefix
2530      'If Left(ScriptEntryArray(I), 1) = "$" Then
2531      'ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 1)
2532      'End If
2533      'Select Case LCase(Left(ScriptEntryArray(I), 2))
2534      'Case "&h", "0x"
2535      'ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 2)
2536      'End Select
2537
2538      'If Left(ScriptEntryArray(I), 1) = "." Then
2539      'ScriptEntryArray(I) = ScriptEntryArray(I).TrimStart(".")

```

```

2539 'If IsNumeric(ScriptEntryArray(I)) Then
2540 'Dim ScriptEntryInt As Integer = Convert.ToInt32(ScriptEntryArray(I))
2541 'ScriptEntryArray(I) = Hex(ScriptEntryInt)
2542 'Else
2543 'Exit For
2544 'End If
2545 'End If
2546
2547 'If IsNumeric("&H" + ScriptEntryArray(I)) Then
2548 ''If IsHexString(ScriptEntryArray(I)) Then
2549 'NumParams = I + 1
2550 'Else
2551 'Exit For
2552 'End If
2553
2554 ''Remove unwanted spaces
2555 'Replace(ScriptEntryArray(I), " ", "")
2556
2557 'Select Case I
2558 'Case 2 'File Offset max. $ffff ffff (dword)
2559 'If Len(ScriptEntryArray(I)) < 8 Then
2560 'ScriptEntryArray(I) = Left("00000000", 8 - Len(ScriptEntryArray(I))) +
ScriptEntryArray(I)
2561 'ElseIf (I = 2) And (len(ScriptEntryArray(I)) > 8) Then
2562 'ScriptEntryArray(I) = Right(ScriptEntryArray(I), 8)
2563 'End If
2564 'Case Else 'File Address, File Length max. $ffff
2565 'If Len(ScriptEntryArray(I)) < 4 Then
2566 'ScriptEntryArray(I) = Left("0000", 4 - Len(ScriptEntryArray(I))) + ScriptEntryArray(I)
2567 'ElseIf Len(ScriptEntryArray(I)) > 4 Then
2568 'ScriptEntryArray(I) = Right(ScriptEntryArray(I), 4)
2569 'End If
2570 'End Select
2571 Next
2572
2573 'Get file variables from script, or get default values if there were none in the script entry
2574 If IO.File.Exists(Replace(FN, "*", "")) = True Then
2575 P = IO.File.ReadAllBytes(Replace(FN, "*", ""))
2576
2577 Select Case NumParams 'ScriptEntryArray.Count
2578 Case 1 'No parameters in script
2579 If InStr(LCase(Replace(FN, "*", "")), ".sid") <> 0 Then 'SID file - read
parameters from file
2580 FA = ConvertIntToHex(P(P(7)) + (P(P(7) + 1) * 256), 4)
2581 FO = ConvertIntToHex(P(7) + 2, 8)
2582 FL = ConvertIntToHex((P.Length - P(7) - 2), 4)
2583 Else 'Any other files
2584 If P.Length > 2 Then 'We have at least 3 bytes
in the file
2585 FA = ConvertIntToHex(P(0) + (P(1) * 256), 4) 'First 2 bytes define load
address
2586 FO = "00000002" 'Offset=2, Length=prg
length-2
2587 FL = ConvertIntToHex(P.Length - 2, 4)
2588 Else 'Short file without
paramters -> STOP
2589 MsgBox("File parameters are needed for the following file:" + vbNewLine +
vbNewLine + FN, vbCritical + vbOKOnly, "Missing file parameters")
2590 GoTo NoDisk
2591 End If
2592 End If
2593 Case 2 'One parameter in script
2594 FA = ScriptEntryArray(1) 'Load address from script

```

```

2595 length=prg length                                'Offset will be 0,
2596         FL = ConvertIntToHex(P.Length, 4)
2597     Case 3 'Two parameters in script
2598         FA = ScriptEntryArray(1)                  'Load address from script
2599         FO = ScriptEntryArray(2)                  'Offset from script
2600         FON = Convert.ToInt32(FO, 16)              'Make sure offset is valid
2601         If FON > P.Length - 1 Then
2602             FON = P.Length - 1                    'If offset>prg length-1
2603         then correct it
2604             FO = ConvertIntToHex(FON, 8)
2605             End If                                'Length=prg length-offset
2606             FL = ConvertIntToHex(P.Length - FON, 4)
2607     Case 4 'Three parameters in script
2608         FA = ScriptEntryArray(1)
2609         FO = ScriptEntryArray(2)
2610         FON = Convert.ToInt32(FO, 16)              'Make sure offset is valid
2611         If FON > P.Length - 1 Then
2612             FON = P.Length - 1                    'If offset>prg length-1
2613         then correct it
2614             FO = ConvertIntToHex(FON, 8)
2615             End If                                'Length=prg length-offset
2616             FL = ScriptEntryArray(3)
2617     End Select
2618     FAN = Convert.ToInt32(FA, 16)
2619     FON = Convert.ToInt32(FO, 16)
2620     FLN = Convert.ToInt32(FL, 16)
2621     'Make sure file length is not longer than actual file (should not happen)
2622     'If FON + FLN > P.Length Then
2623     'FLN = P.Length - FON
2624     'End If
2625     'Make sure file address+length<=&H10000
2626     If FAN + FLN > &H10000 Then
2627         FLN = (&H10000 - FAN) And &HF00
2628         If FLN < &H100 Then
2629             MsgBox("The Hi-Score File's size must be at least $100 bytes!", vbOKOnly +
2630 vbExclamation, "Hi-Score File Error")
2631             GoTo NoDisk
2632         End If
2633     End If
2634     'Round UP to nearest $100, at least $100 but not more than $0f00 bytes
2635     FLN = If((FLN Mod &H100 <> 0) Or (FLN = 0), FLN + &H100, FLN) And &HF00
2636     FL = ConvertIntToHex(FLN, 4)
2637     'Trim file to the specified chunk (FLN number of bytes starting at FON, to Address of FAN)
2638     Dim PL As List(Of Byte) = P.ToList              'Copy array to list
2639     P = PL.Skip(FON).Take(FLN).ToArray              'Trim file to specified segment (FLN number of
2640 bytes starting at FON)
2641     If P.Length < FLN Then
2642         ReDim Preserve P(FLN - 1)                  'Round length up to nearest $100
2643     End If
2644     HSFile = P
2645     HSFileName = FN
2646     HSAddress = FAN
2647     HSOffset = FON
2648     HSLength = FLN

```

```

2653         bSaverPlugin = True
2654
2655     Else
2656         'Add code here to create blank HSFile here if all 3 parameters are present
2657     If ScriptEntryArray.Count = 4 Then
2658         FA = ScriptEntryArray(1)
2659         FO = ScriptEntryArray(2)
2660         FL = ScriptEntryArray(3)
2661
2662         FAN = Convert.ToInt32(FA, 16)
2663         FON = Convert.ToInt32(FO, 16)
2664         FLN = Convert.ToInt32(FL, 16)
2665
2666         'Make sure file address+length<=&H10000
2667     If FAN + FLN > &H10000 Then
2668         FLN = (&H10000 - FAN) And &HF00
2669         If FLN < &H100 Then
2670             MsgBox("The Hi-Score File's size must be at least $100 bytes!", vbOKOnly +
2671 vbExclamation, "Hi-Score File Error")
2672             GoTo NoDisk
2673         End If
2674     End If
2675
2676     'Round UP to nearest $100, at least $100 but not more than $0f00 bytes
2677     FLN = If((FLN Mod &H100 <> 0) Or (FLN = 0), FLN + &H100, FLN) And &HF00
2678
2679     FL = ConvertIntToHex(FLN, 4)
2680
2681     ReDim P(FLN - 1) 'Create blank HSFile based on file parameters
2682
2683     HSFile = P
2684     HSFileName = FN
2685     HSAddress = FAN
2686     HSOffset = FON
2687     HSLength = FLN
2688
2689     bSaverPlugin = True
2690
2691     Else
2692         MsgBox("The following Hi-Score File does not exist:" + vbNewLine + vbNewLine + FN,
2693 vbOKOnly + vbCritical, "Hi-Score File not found")
2694         GoTo NoDisk
2695     End If
2696 End If
2697
2698 Exit Function
2699 Err:
2700     ErrCode = Err.Number
2701     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
2702 + " Error")
2703 NoDisk:
2704     AddHSFile = False
2705 End Function
2706
2707 Private Function ParameterIsNumeric(I As Integer) As Boolean
2708     If DoOnError Then On Error GoTo Err
2709
2710     'Remove unwanted spaces
2711     ScriptEntryArray(I) = Replace(ScriptEntryArray(I), " ", "")

```

```

2712 'Remove HEX prefix
2713 If Left(ScriptEntryArray(I), 1) = "$" Then
2714     ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 1)
2715 End If
2716
2717 Select Case LCase(Left(ScriptEntryArray(I), 2))
2718     Case "&h", "0x"
2719         ScriptEntryArray(I) = Right(ScriptEntryArray(I), Len(ScriptEntryArray(I)) - 2)
2720 End Select
2721
2722 'If decimal -> convert it to hex
2723 If Left(ScriptEntryArray(I), 1) = "." Then
2724     ScriptEntryArray(I) = ScriptEntryArray(I).TrimStart(".")
2725     If IsNumeric(ScriptEntryArray(I)) Then
2726         Dim ScriptEntryInt As Integer = Convert.ToInt32(ScriptEntryArray(I))
2727         ScriptEntryArray(I) = Hex(ScriptEntryInt)
2728     Else
2729         ParameterIsNumeric = False
2730         Exit Function
2731     End If
2732 End If
2733
2734 ParameterIsNumeric = IsNumeric("&H" + ScriptEntryArray(I))
2735
2736 Exit Function
2737
2738 Err:
2739     ErrCode = Err.Number
2740     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
2741
2742
2743 End Function
2744
2745 Private Sub CorrectParameterStringLength(I As Integer)
2746     If DoOnError Then On Error GoTo Err
2747
2748     Select Case I
2749         Case 2 'File Offset max. $ffff ffff (dword)
2750             If Len(ScriptEntryArray(I)) < 8 Then
2751                 ScriptEntryArray(I) = Left("00000000", 8 - Len(ScriptEntryArray(I))) +
ScriptEntryArray(I)
2752             ElseIf (I = 2) And (Len(ScriptEntryArray(I)) > 8) Then
2753                 ScriptEntryArray(I) = Right(ScriptEntryArray(I), 8)
2754             End If
2755         Case Else 'File Address, File Length max. $ffff
2756             If Len(ScriptEntryArray(I)) < 4 Then
2757                 ScriptEntryArray(I) = Left("0000", 4 - Len(ScriptEntryArray(I))) +
ScriptEntryArray(I)
2758             ElseIf Len(ScriptEntryArray(I)) > 4 Then
2759                 ScriptEntryArray(I) = Right(ScriptEntryArray(I), 4)
2760             End If
2761     End Select
2762
2763 Exit Sub
2764 Err:
2765     ErrCode = Err.Number
2766     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
2767
2768 End Sub
2769
2770 Public Function AddVirtualFile() As Boolean

```

```

2771 If DoOnErr Then On Error GoTo Err
2772
2773 AddVirtualFile = True
2774
2775 If NewBundle = True Then
2776     NewBundle = False
2777     If BundleDone() = False Then GoTo NoDisk
2778 End If
2779
2780 Dim FN As String = ScriptEntryArray(0)
2781 Dim FA As String = ""
2782 Dim FO As String = ""
2783 Dim FL As String = ""
2784 Dim FAN As Integer
2785 Dim FON As Integer
2786 Dim FLN As Integer
2787 Dim FUIO As Boolean = False
2788
2789 Dim NumParams As Integer = 1
2790
2791 Dim P() As Byte
2792
2793 If Right(FN, 1) = "*" Then
2794     FN = Replace(FN, "*", "")
2795     FUIO = True
2796 End If
2797
2798 If InStr(FN, ":") = 0 Then          'relative file path
2799     FN = ScriptPath + FN          'look for file in script's folder
2800 End If
2801
2802 'Correct file parameter lengths to 4-8 characters
2803 For I As Integer = 1 To ScriptEntryArray.Count - 1
2804
2805     If ParameterIsNumeric(I) Then
2806         NumParams += 1
2807     Else
2808         Exit For
2809     End If
2810
2811     CorrectParameterStringLength(I)
2812
2813 Next
2814
2815 'Get file variables from script, or get default values if there were none in the script entry
2816 If IO.File.Exists(FN) = True Then
2817     P = IO.File.ReadAllBytes(FN)
2818
2819     Select Case NumParams 'ScriptEntryArray.Count
2820     Case 1 'No parameters in script
2821         If InStr(LCase(FN), ".sid") <> 0 Then 'SID file - read parameters from file
2822             FA = ConvertIntToHex(P(P(7)) + (P(P(7) + 1) * 256), 4)
2823             FO = ConvertIntToHex(P(P(7) + 2), 8)
2824             FL = ConvertIntToHex((P.Length - P(7) - 2), 4)
2825         Else 'Any other files
2826             If P.Length > 2 Then 'We have at least 3 bytes
2827                 FA = ConvertIntToHex(P(0) + (P(1) * 256), 4) 'First 2 bytes define
2828                 FO = "00000002" 'Offset=2, Length=prg
2829                 FL = ConvertIntToHex(P.Length - 2, 4)

```



```

2830 Else 'Short file without
parameters -> STOP
2831 MsgBox("File parameters are needed for the following file:" + vbNewLine +
vbNewLine + FN, vbCritical + vbOKOnly, "Missing file parameters")
2832 GoTo NoDisk
2833 End If
2834 End If
2835 Case 2 'One parameter in script
2836 FA = ScriptEntryArray(1) 'Load address from script
2837 FO = "00000000" 'Offset will be 0,
length=prg length
2838 FL = ConvertIntToHex(P.Length, 4)
2839 Case 3 'Two parameters in script
2840 FA = ScriptEntryArray(1) 'Load address from script
2841 FO = ScriptEntryArray(2) 'Offset from script
2842 FON = Convert.ToInt32(FO, 16) 'Make sure offset is valid
2843 If FON > P.Length - 1 Then
2844 FON = P.Length - 1 'If offset>prg length-1
then correct it
2845 FO = ConvertIntToHex(FON, 8)
2846 End If 'Length=prg length- offset
2847 FL = ConvertIntToHex(P.Length - FON, 4)
2848 Case 4 'Three parameters in script
2849 FA = ScriptEntryArray(1)
2850 FO = ScriptEntryArray(2)
2851 FON = Convert.ToInt32(FO, 16) 'Make sure offset is valid
2852 If FON > P.Length - 1 Then
2853 MsgBox("Invalid offset detected in the following entry:" + vbNewLine +
vbNewLine +
2854 ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid
offset")
2855 GoTo NoDisk
2856 'FON = P.Length - 1 'If offset>prg length-1
then correct it
2857 'FO = ConvertIntToHex(FON, 8)
2858 End If 'Length=prg length- offset
2859 FL = ScriptEntryArray(3)
2860 End Select
2861
2862 FAN = Convert.ToInt32(FA, 16)
2863 FON = Convert.ToInt32(FO, 16)
2864 FLN = Convert.ToInt32(FL, 16)
2865
2866 'Make sure file length is not longer than actual file (should not happen)
2867 If FON + FLN > P.Length Then
2868 MsgBox("Invalid file length detected in the following entry:" + vbNewLine + vbNewLine
+
2869 ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid
virtual file length")
2870 GoTo NoDisk
2871 'FLN = P.Length - FON
2872 'FL = ConvertIntToHex(FLN, 4)
2873 End If
2874
2875 'Make sure file address+length<=&H10000
2876 If FAN + FLN > &H10000 Then
2877 MsgBox("Invalid file address and/or length detected in the following entry:" +
vbNewLine + vbNewLine +
2878 ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid
virtual file address and/or length")
2879 GoTo NoDisk
2880 'FLN = &H10000 - FAN
2881 'FL = ConvertIntToHex(FLN, 4)
2882 End If

```

```

2883         'Trim file to the specified chunk (FLN number of bytes starting at FON, to Address of FAN)
2884         Dim PL As List(Of Byte) = P.ToList          'Copy array to list
2885         P = PL.Skip(FON).Take(FLN).ToArray          'Trim file to specified segment (FLN number of
bytes starting at FON)
2887
2888         Else
2889
2890             MsgBox("The following file does not exist:" + vbNewLine + vbNewLine + FN, vbOKOnly +
vbCritical, "File not found")
2891             GoTo NoDisk
2892
2893         End If
2894
2895         VFileCnt += 1
2896         ReDim Preserve tmpVFileNameA(VFileCnt), tmpVFileAddrA(VFileCnt), tmpVFileOffsA(VFileCnt),
tmpVFileLenA(VFileCnt), tmpVFileIOA(VFileCnt)
2897
2898         tmpVFileNameA(VFileCnt) = ScriptEntryArray(0) 'FN
2899         tmpVFileAddrA(VFileCnt) = FA
2900         tmpVFileOffsA(VFileCnt) = FO          'This may not be needed later
2901         tmpVFileLenA(VFileCnt) = FL
2902         tmpVFileIOA(VFileCnt) = FUIO
2903
2904         tmpVFiles.Add(P)
2905
2906         Exit Function
2907 Err:
2908         ErrCode = Err.Number
2909         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
2910 NoDisk:
2911         AddVirtualFile = False
2912     End Function
2913
2914     Public Function AddFileToBundle() As Boolean
2915         If DoOnErr Then On Error GoTo Err
2916
2917         AddFileToBundle = True
2918
2919         Dim FN As String = ScriptEntryArray(0)
2920         Dim FA As String = ""
2921         Dim FO As String = ""
2922         Dim FL As String = ""
2923         Dim FAN As Integer
2924         Dim FON As Integer
2925         Dim FLN As Integer
2926         Dim FUIO As Boolean = False
2927
2928         Dim NumParams As Integer = 1
2929
2930         Dim P() As Byte
2931
2932         If Right(FN, 1) = "*" Then
2933             FN = Replace(FN, "*", "")
2934             FUIO = True
2935         End If
2936
2937         If InStr(FN, ":") = 0 Then 'relative file path
2938             FN = ScriptPath + FN          'look for file in script's folder
2939         End If
2940
2941         '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

```

3003 If InStr(LCase(FN), ".sid") <> 0 Then 'SID file - read parameters from file
3004     FA = ConvertIntToHex(P(P(7)) + (P(P(7) + 1) * 256), 4)
3005     FO = ConvertIntToHex(P(7) + 2, 8)
3006     FL = ConvertIntToHex((P.Length - P(7) - 2), 4)
3007 Else 'Any other files
3008     If P.Length > 2 Then 'We have at least 3 bytes
in the file
3009         FA = ConvertIntToHex(P(0) + (P(1) * 256), 4) 'First 2 bytes define
load address
3010         FO = "00000002" 'Offset=2, Length=prg
length-2
3011         FL = ConvertIntToHex(P.Length - 2, 4)
3012     Else 'Short file without
paramters -> STOP
3013         MsgBox("File parameters are needed for the following file:" + vbNewLine +
vbNewLine + FN, vbCritical + vbOKOnly, "Missing file parameters")
3014         GoTo NoDisk
3015     End If
3016 End If
3017 Case 2 'One parameter in script
3018     FA = ScriptEntryArray(1) 'Load address from script
3019     FO = "00000000" 'Offset will be 0,
length=prg length
3020     FL = ConvertIntToHex(P.Length, 4)
3021 Case 3 'Two parameters in script
3022     FA = ScriptEntryArray(1) 'Load address from script
3023     FO = ScriptEntryArray(2) 'Offset from script
3024     FON = Convert.ToInt32(FO, 16) 'Make sure offset is valid
3025     If FON > P.Length - 1 Then
3026         FON = P.Length - 1 'If offset>prg length-1
then correct it
3027         FO = ConvertIntToHex(FON, 8)
3028     End If 'Length=prg length- offset
3029     FL = ConvertIntToHex(P.Length - FON, 4)
3030 Case 4 'Three parameters in script
3031     FA = ScriptEntryArray(1)
3032     FO = ScriptEntryArray(2)
3033     FON = Convert.ToInt32(FO, 16) 'Make sure offset is valid
3034     If FON > P.Length - 1 Then
3035         MsgBox("Invalid offset detected in the following entry:" + vbNewLine +
vbNewLine +
3036         ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid
offset")
3037         GoTo NoDisk
3038         'FON = P.Length - 1 'If offset>prg length-1
then correct it
3039         'FO = ConvertIntToHex(FON, 8)
3040     End If 'Length=prg length- offset
3041     FL = ScriptEntryArray(3)
3042 End Select
3043
3044 FAN = Convert.ToInt32(FA, 16)
3045 FON = Convert.ToInt32(FO, 16)
3046 FLN = Convert.ToInt32(FL, 16)
3047
3048 'Make sure file length is not longer than actual file (should not happen)
3049 If FON + FLN > P.Length Then
3050     MsgBox("Invalid file length detected in the following entry:" + vbNewLine + vbNewLine
+
3051     ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid
file length")
3052     GoTo NoDisk
3053     'FLN = P.Length - FON
3054     'FL = ConvertIntToHex(FLN, 4)

```

```

3055         End If
3056
3057         'Make sure file address+length<=&H10000
3058         If FAN + FLN > &H10000 Then
3059             MsgBox("Invalid file address and/or length detected in the following entry:" +
vbNewLine + vbNewLine +
3060                 ScriptEntryType + vbTab + ScriptEntry, vbOKOnly + vbCritical, "Invalid
file address and/or length")
3061             GoTo NoDisk
3062             'FLN = &H10000 - FAN
3063             'FL = ConvertIntToHex(FLN, 4)
3064         End If
3065
3066         'Trim file to the specified chunk (FLN number of bytes starting at FON, to Address of FAN)
3067         Dim PL As List(Of Byte) = P.ToList           'Copy array to list
3068         P = PL.Skip(FON).Take(FLN).ToArray         'Trim file to specified segment (FLN number of
bytes starting at FON)
3069
3070     Else
3071
3072         MsgBox("The following file does not exist:" + vbNewLine + vbNewLine + FN, vbOKOnly +
vbCritical, "File not found")
3073         GoTo NoDisk
3074
3075     End If
3076
3077     FileCnt += 1
3078     ReDim Preserve tmpFileNameA(FileCnt), tmpFileAddrA(FileCnt), tmpFileOffsA(FileCnt),
tmpFileLenA(FileCnt), tmpFileIOA(FileCnt)
3079
3080     tmpFileNameA(FileCnt) = FN
3081     tmpFileAddrA(FileCnt) = FA
3082     tmpFileOffsA(FileCnt) = FO           'This may not be needed later
3083     tmpFileLenA(FileCnt) = FL
3084     tmpFileIOA(FileCnt) = FUIO
3085
3086     UncompBundleSize += Int(FLN / 256)
3087     If FLN Mod 256 <> 0 Then
3088         UncompBundleSize += 1
3089     End If
3090
3091     If FirstFileOfDisk = True Then           'If Demo Start is not specified, we will use the start
address of the first file
3092         FirstFileStart = FA
3093         FirstFileOfDisk = False
3094     End If
3095
3096     tmpPrgs.Add(P)
3097
3098     Exit Function
3099 Err:
3100     ErrCode = Err.Number
3101     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3102 NoDisk:
3103     AddFileToBundle = False
3104
3105 End Function
3106
3107 Private Function SplitScriptEntry() As Boolean
3108     If DoOnError Then On Error GoTo Err
3109
3110     SplitScriptEntry = True

```

```

3111
3112     If InStr(ScriptEntry, vbTab) = 0 Then
3113         ScriptEntryType = Replace(ScriptEntry, " ", "")
3114         ScriptEntry = ""
3115     Else
3116         ScriptEntryType = Replace(Left(ScriptEntry, InStr(ScriptEntry, vbTab) - 1), " ", "")
3117         ScriptEntry = Right(ScriptEntry, Len(ScriptEntry) - InStr(ScriptEntry, vbTab))
3118     End If
3119
3120     LastNonEmpty = -1
3121
3122     ReDim ScriptEntryArray(LastNonEmpty)
3123
3124     If ScriptEntry = "" Then Exit Function
3125
3126     ScriptEntryArray = Split(ScriptEntry, vbTab)
3127
3128     For I As Integer = 0 To ScriptEntryArray.Length - 1
3129         If ScriptEntryArray(I) <> "" Then
3130             LastNonEmpty += 1
3131             ScriptEntryArray(LastNonEmpty) = ScriptEntryArray(I)
3132         End If
3133     Next
3134
3135     If LastNonEmpty > -1 Then
3136         ReDim Preserve ScriptEntryArray(LastNonEmpty)
3137     Else
3138         ReDim ScriptEntryArray(0)
3139     End If
3140
3141     Exit Function
3142 Err:
3143     ErrCode = Err.Number
3144     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3145
3146     SplitScriptEntry = False
3147
3148 End Function
3149
3150 Public Function ResetDiskVariables() As Boolean
3151     If DoOnError Then On Error GoTo Err
3152
3153     ResetDiskVariables = True
3154
3155     If DiskCnt = 126 Then
3156         MsgBox("You have reached the maximum number of disks in this project!", vbOKOnly +
vbInformation, "Can't have more than 127 disks :)")
3157         GoTo NoDisk
3158     End If
3159
3160     DiskCnt += 1
3161     ReDim Preserve DiskSizeA(DiskCnt)
3162     'Reset Bundle File variables here, to have an empty array for the first compression on a
ReBuild
3163     Prgs.Clear() 'this is the one that is needed for the first CompressPart call during a
ReBuild
3164     ReDim FileNameA(-1), FileAddrA(-1), FileOffsA(-1), FileLenA(-1), FileIOA(-1) 'but reset all
arrays just to be safe
3165
3166     'Reset directory arrays
3167     ReDim DirBlocks(511), DirPtr(127)
3168

```

```

3169 'Reset disk system to support 35 tracks
3170 TracksPerDisk = 35
3171
3172 'Reset Hi-Score Saver plugin variables
3173 bSaverPlugin = False
3174 HSFileName = ""
3175 HSAddress = 0
3176 HSOffset = 0
3177 HSLength = 0
3178
3179 'Reset interleave
3180 ResetInterleaves()
3181
3182 BufferCnt = 0
3183 BundleNo = 0
3184 MaxBundleNoExceeded = False
3185
3186 ReDim ByteSt(-1)
3187 ResetBuffer()
3188
3189
3190 D64Name = ""
3191 DiskHeader = "" '"demo disk " + Year(Now).ToString
3192 DiskID = "" '"sprkl"
3193 DemoName = "" '"demo"
3194 DemoStart = ""
3195 DirArtName = ""
3196 LoaderZP = "02"
3197
3198 'Reset Disk image
3199 NewDisk()
3200
3201 BlockPtr = 1
3202
3203 '-----
3204
3205 StartTrack = 1 : StartSector = 0
3206
3207 NextTrack = StartTrack
3208 NextSector = StartSector
3209
3210 BitsSaved = 0 : BytesSaved = 0
3211
3212 FirstFileOfDisk = True 'To save Start Address of first file on disk if Demo Start is not
specified
3213
3214 '-----
3215
3216 BundleCnt = -1 'WILL BE INCREASED TO 0 IN ResetPartVariables
3217 LoaderBundles = 1
3218 FilesInBuffer = 1
3219
3220 CurrentBundle = -1
3221
3222 '-----
3223
3224 If ResetBundleVariables() = False Then GoTo NoDisk 'Also adds first bundle
3225
3226 NewBundle = False
3227
3228 Exit Function
3229 Err:

```

```

3230         ErrCode = Err.Number
3231         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3232 NoDisk:
3233         ResetDiskVariables = False
3234
3235     End Function
3236
3237     Public Sub ResetInterleaves()
3238         If DoOnError Then On Error GoTo Err
3239
3240         IL0 = DefaultIL0
3241         IL1 = DefaultIL1
3242         IL2 = DefaultIL2
3243         IL3 = DefaultIL3
3244
3245         Exit Sub
3246
3247 Err:
3248         ErrCode = Err.Number
3249         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3250     End Sub
3251
3252     Public Function ResetBundleVariables() As Boolean
3253         If DoOnError Then On Error GoTo Err
3254
3255         ResetBundleVariables = True
3256
3257         FileCnt = -1
3258         ReDim tmpFileNameA(FileCnt), tmpFileAddrA(FileCnt), tmpFileOffsA(FileCnt),
tmpFileLenA(FileCnt), tmpFileIOA(FileCnt)
3259
3260         VFileCnt = -1
3261         ReDim tmpVFileNameA(VFileCnt), tmpVFileAddrA(VFileCnt), tmpVFileOffsA(VFileCnt),
tmpVFileLenA(VFileCnt), tmpVFileIOA(VFileCnt)
3262
3263         tmpVFiles.Clear()
3264
3265         tmpPrgs.Clear()
3266
3267         BundleCnt += 1
3268
3269         TotalBundles += 1
3270         ReDim Preserve BundleSizeA(TotalBundles), BundleOrigSizeA(TotalBundles)
3271         BlockCnt = 0
3272
3273         UncompBundleSize = 0
3274
3275         Exit Function
3276 Err:
3277         ErrCode = Err.Number
3278         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3279
3280         ResetBundleVariables = False
3281
3282     End Function
3283
3284     Public Function AddCompressedBundlesToDisk() As Boolean
3285         If DoOnError Then On Error GoTo Err
3286
3287         AddCompressedBundlesToDisk = True

```



```

3288         If BlocksFree < BufferCnt Then
3290             MsgBox(D64Name + " cannot be built because it would require " + BufferCnt.ToString + "
blocks." + vbNewLine + vbNewLine +
3291                 "This disk only has " + SectorsPerDisk.ToString + " blocks.", vbOKOnly + vbCritical,
"Not enough free space on disk")
3292             GoTo NoDisk
3293         End If
3294
3295         CalcILTab()
3296
3297         InjectDirBlocks()
3298
3299         For I = 0 To BufferCnt - 1
3300             CT = TabT(I)
3301             CS = TabS(I)
3302             For J = 0 To 255
3303                 Disk(Track(CT) + 256 * CS + J) = ByteSt(I * 256 + J)
3304             Next
3305
3306             DeleteBit(CT, CS, True)
3307         Next
3308
3309         If BufferCnt < SectorsPerDisk Then
3310             NextTrack = TabT(BufferCnt)
3311             NextSector = TabS(BufferCnt)
3312         Else
3313             NextTrack = 18
3314             NextSector = 0
3315         End If
3316
3317         Exit Function
3318 Err:
3319         ErrCode = Err.Number
3320         MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3321 NoDisk:
3322         AddCompressedBundlesToDisk = False
3323
3324     End Function
3325
3326     Public Function AddDirArt() As Boolean
3327         If DoOnErr Then On Error GoTo Err
3328
3329         AddDirArt = True
3330
3331         'Make sure strings have values
3332         If DirArtName Is Nothing Then DirArtName = ""
3333         If DirArt Is Nothing Then DirArt = ""
3334
3335         If DirArtName = "" Then Exit Function
3336
3337         If IO.File.Exists(DirArtName) = False Then
3338             MsgBox("The following DirArt file does not exist: " + vbNewLine + vbNewLine + DirArtName,
vbOKOnly + vbExclamation, "DirArt file cannot be found")
3339             GoTo NoDisk
3340         End If
3341
3342         Dim DAN() As String = DirArtName.Split(".")
3343
3344         Dim DirArtType As String = ""
3345
3346         If DAN.Length > 1 Then

```

```

3347         DirArtType = DAN(DAN.Length - 1)
3348     End If
3349
3350     Select Case LCase(DirArtType)
3351     Case "d64"
3352         ConvertD64ToDirArt()
3353     Case "txt"
3354         ConvertTxtToDirArt()
3355     Case "prg"
3356         ConvertBintoDirArt(LCase(DirArtType))
3357     Case Else
3358         ConvertBintoDirArt()
3359     End Select
3360
3361     Exit Function
3362 Err:
3363     ErrCode = Err.Number
3364     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3365 NoDisk:
3366     AddDirArt = False
3367
3368 End Function
3369
3370 Private Sub ConvertBintoDirArt(Optional DirArtType As String = "bin")
3371     If DoOnError Then On Error GoTo Err
3372
3373     Dim DA() As Byte = IO.File.ReadAllBytes(DirArtName)
3374
3375     DirTrack = 18
3376     DirSector = 1
3377     Dim NB As Byte = 0
3378     Dim DAPtr As Integer = If(DirArtType = "prg", 2, 0)
3379 NextSector:
3380     For B As Integer = DAPtr To DA.Length - 1 Step 40
3381
3382         FindNextDirPos()
3383
3384         If DirPos <> 0 Then
3385             Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 0) = &H82    '"PRG" - all dir
entries will point at first file in dir
3386             Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 1) = 18    'Track 18 (track
pointer of boot loader)
3387             Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 2) = 7    'Sector 7 (sector
pointer of boot loader)
3388
3389             For I As Integer = 0 To 15
3390                 If B + I < DA.Length Then
3391                     Select Case DA(B + I)
3392                     Case 0 To 31
3393                         NB = DA(B + I) + 64
3394                     Case 32 To 63
3395                         NB = DA(B + I)
3396                     Case 64 To 95
3397                         NB = DA(B + I) + 128
3398                     Case 96 To 127
3399                         NB = DA(B + I) + 64
3400                     Case 128 To 159
3401                         NB = DA(B + I) - 128
3402                     Case 160 To 191
3403                         NB = DA(B + I) - 64
3404                     Case 192 To 223

```

```

3405         NB = DA(B + I) - 64
3406         Case 224 To 254
3407             NB = DA(B + I)
3408         End Select
3409         Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 3 + I) = NB
3410     Else
3411         Exit For
3412     End If
3413 Next
3414 If (DirTrack = 18) AndAlso (DirSector = 1) AndAlso (DirPos = 2) Then
3415     'Very first dir entry, also add loader block count
3416     Disk(Track(DirTrack) + (DirSector * 256) + DirPos + &H1C) = LoaderBlockCount
3417 End If
3418 Else
3419     Exit For
3420 End If
3421 Next
3422
3423 Exit Sub
3424 Err:
3425     ErrCode = Err.Number
3426     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3427
3428 End Sub
3429
3430 Private Sub ConvertD64ToDirArt()
3431     If DoOnError Then On Error GoTo Err
3432
3433     Dim DA() As Byte = IO.File.ReadAllBytes(DirArtName)
3434
3435     Dim T As Integer = 18
3436     Dim S As Integer = 1
3437
3438     DirTrack = 18
3439     DirSector = 1
3440     Dim DirFull As Boolean = False
3441 NextSector:
3442     Dim DAPtr As Integer = Track(T) + (S * 256)
3443     For B As Integer = 2 To 255 Step 32
3444         If DA(DAPtr + B) <> 0 Then             '= &H82 Then             'PRG file type
3445
3446             FindNextDirPos()
3447
3448             If DirPos <> 0 Then
3449                 Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 0) = &H82      '"PRG" - all dir
entries will point at first file in dir
3450                 Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 1) = 18      'Track 18 (track
pointer of boot loader)
3451                 Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 2) = 7      'Sector 7 (sector
pointer of boot loader)
3452
3453                 For I As Integer = 0 To 15
3454                     Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 3 + I) = DA(DAPtr + B + 3
+ I)
3455                 Next
3456
3457                 If (DirTrack = 18) AndAlso (DirSector = 1) AndAlso (DirPos = 2) Then
3458                     'Very first dir entry, also add loader block count
3459                     Disk(Track(DirTrack) + (DirSector * 256) + DirPos + &H1C) = LoaderBlockCount
3460                 End If
3461             Else
3462                 DirFull = True

```

```

3463         Exit For
3464     End If
3465 End If
3466 Next
3467
3468 If (DirFull = False) And (DA(DAPtr) <> 0) Then
3469     T = DA(DAPtr)
3470     S = DA(DAPtr + 1)
3471     GoTo NextSector
3472 End If
3473
3474 If DiskHeader = "" Then
3475     For I As Integer = 0 To 15
3476         Disk(Track(18) + &H90 + I) = DA(Track(18) + &H90 + I)
3477     Next
3478 End If
3479
3480 If DiskID = "" Then
3481     For I As Integer = 0 To 4
3482         Disk(Track(18) + &HA2 + I) = DA(Track(18) + &HA2 + I)
3483     Next
3484 End If
3485
3486 Exit Sub
3487 Err:
3488     ErrCode = Err.Number
3489     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3490
3491 End Sub
3492
3493 Private Sub ConvertTxtToDirArt()
3494     If DoOnError Then On Error GoTo Err
3495
3496     DirArt = IO.File.ReadAllText(DirArtName)
3497
3498     Dim DirEntries() As String = DirArt.Split(vbLf)
3499
3500     DirTrack = 18
3501     DirSector = 1
3502     For I As Integer = 0 To DirEntries.Count - 1
3503         DirEntry = DirEntries(I).TrimEnd(Chr(13))
3504         FindNextDirPos()
3505         If DirPos <> 0 Then
3506             AddDirEntry()
3507         Else
3508             Exit For
3509         End If
3510     Next
3511
3512 Exit Sub
3513 Err:
3514     ErrCode = Err.Number
3515     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3516
3517 End Sub
3518
3519 Private Sub AddDirEntry()
3520     If DoOnError Then On Error GoTo Err
3521
3522     Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 0) = &H82  "'PRG" - all dir entries will

```

```

point at first file in dir
3523     Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 1) = 18      'Track 18 (track pointer of
boot loader)
3524     Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 2) = 7      'Sector 7 (sector pointer of
boot loader)
3525
3526     'Remove vbNewLine characters and add 16 SHIFT+SPACE tail characters
3527     DirEntry += StrDup(16, Chr(160))
3528
3529     'Copy only the first 16 characters of the edited DirEntry to the Disk Directory
3530     For I As Integer = 1 To 16
3531         Disk(Track(DirTrack) + (DirSector * 256) + DirPos + 2 + I) = Asc(Mid(UCCase(DirEntry), I,
1))
3532     Next
3533
3534     If (DirTrack = 18) AndAlso (DirSector = 1) AndAlso (DirPos = 2) Then
3535         'Very first dir entry, also add loader block count
3536         Disk(Track(DirTrack) + (DirSector * 256) + DirPos + &H1C) = LoaderBlockCount
3537     End If
3538
3539     'Reset DirEntry
3540     DirEntry = ""
3541
3542     Exit Sub
3543 Err:
3544     ErrCode = Err.Number
3545     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3546
3547     End Sub
3548
3549     Public Sub FindNextDirPos()
3550         If DoOnError Then On Error GoTo Err
3551
3552         DirPos = 0
3553
3554 FindNextEntry:
3555
3556         For I As Integer = 2 To 255 Step 32
3557             If Disk(Track(DirTrack) + (DirSector * 256) + I) = 0 Then
3558                 DirPos = I
3559                 Exit Sub
3560             End If
3561         Next
3562
3563         FindNextDirSector()
3564
3565         If DirSector <> 0 Then GoTo FindNextEntry
3566
3567         Exit Sub
3568 Err:
3569     ErrCode = Err.Number
3570     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3571
3572     End Sub
3573
3574     Private Sub FindNextDirSector()
3575         If DoOnError Then On Error GoTo Err
3576
3577         'Sector order: 1,7,13,3,9,15,4,8,12,16
3578
3579         LastDirSector = DirSector

```

```
3580
3581     If DirSector < 6 Then
3582         DirSector += 1
3583     Else
3584         DirSector = 0
3585     End If
3586
3587     'Select Case DirSector
3588     'Case 1
3589     'DirSector = 2
3590     'Case 7
3591     'DirSector = 13
3592     'Case 13
3593     'DirSector = 3
3594     'Case 3
3595     'DirSector = 9
3596     'Case 9
3597     'DirSector = 15
3598     'Case 15
3599     'DirSector = 4
3600     'Case 4
3601     'DirSector = 8
3602     'Case 8
3603     'DirSector = 12
3604     'Case 12
3605     'DirSector = 16
3606     'Case 16
3607     'DirSector = 0
3608     'End Select
3609
3610     Disk(Track(DirTrack) + (LastDirSector * 256)) = DirTrack
3611     Disk(Track(DirTrack) + (LastDirSector * 256) + 1) = DirSector
3612
3613     If DirSector <> 0 Then
3614         Disk(Track(DirTrack) + (DirSector * 256)) = 0
3615         Disk(Track(DirTrack) + (DirSector * 256) + 1) = 255
3616     End If
3617
3618     Exit Sub
3619 Err:
3620     ErrCode = Err.Number
3621     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
3622 + " Error")
3623
3624 End Sub
3625
3626 Public Sub SetScriptPath(Path As String)
3627     If DoOnError Then On Error GoTo Err
3628
3629     ScriptName = Path
3630     ScriptPath = ScriptName
3631
3632     If Path = "" Then Exit Sub
3633
3634     For I As Integer = Len(Path) - 1 To 0 Step -1
3635         If Right(ScriptPath, 1) <> "\" Then
3636             ScriptPath = Left(ScriptPath, Len(ScriptPath) - 1)
3637         Else
3638             Exit For
3639         End If
3640     Next
```

```

3641 Exit Sub
3642 Err:
3643     ErrCode = Err.Number
3644     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3645
3646 End Sub
3647
3648 Public Sub CalcILTab()
3649     If DoOnError Then On Error GoTo Err
3650
3651     Dim SMax, IL As Integer
3652     Dim Disk(682 + 85) As Byte
3653     Dim I As Integer = 0
3654     Dim SCnt As Integer
3655     Dim Tr(40) As Integer
3656     Dim S As Integer = 0
3657     Dim LastS As Integer = 0
3658
3659     Tr(1) = 0
3660     For T = 1 To TracksPerDisk - 1 '34
3661         Select Case T
3662             Case 1 To 17
3663                 Tr(T + 1) = Tr(T) + 21
3664             Case 18 To 24
3665                 Tr(T + 1) = Tr(T) + 19
3666             Case 25 To 30
3667                 Tr(T + 1) = Tr(T) + 18
3668             Case 31 To 40
3669                 Tr(T + 1) = Tr(T) + 17
3670         End Select
3671     Next
3672
3673     For T As Integer = 1 To TracksPerDisk
3674         TabStartS(T) = 255
3675         If T = 18 Then
3676             T += 1
3677             TabStartS(T) = 255
3678             S += 2
3679         End If
3680
3681         SCnt = 0
3682
3683         Select Case T
3684             Case 1 To 17
3685                 SMax = 21
3686                 IL = IL0
3687             Case 18 To 24
3688                 SMax = 19
3689                 IL = IL1
3690             Case 25 To 30
3691                 SMax = 18
3692                 IL = IL2
3693             Case 31 To 40
3694                 SMax = 17
3695                 IL = IL3
3696         End Select
3697
3698         IL = IL Mod SMax
3699
3700         'If SectorSkew <> 0 Then
3701         'If T = 19 Then

```

```

3702 'S = LastS - ((2 * SectorSkew) + 4) 'Extra sector skew for skipping track 18
3703 'ElseIf T <> 1 Then
3704 'S = LastS - SectorSkew 'Sector Skew
3705 'End If
3706 'If S < 0 Then
3707 'S += SMax
3708 'End If
3709 'Else
3710 'If T = 18 Then
3711 'S += 2
3712 'End If
3713 'End If
3714
3715 'S = 0 'Reset first sector for each track
3716
3717 GoTo NextStart
3718
3719 NextSector:
3720 If Disk(Tr(T) + S) = 0 Then
3721 Disk(Tr(T) + S) = 1
3722 TabT(I) = T
3723 TabS(I) = S
3724 LastS = S
3725 TabSCnt(I) = SMax - SCnt
3726 I += 1
3727 SCnt += 1
3728 S += IL
3729 NextStart:
3730 If S >= SMax Then
3731 S -= SMax
3732 If (T < 18) And (S > 0) Then S -= 1 'Wrap around: Subtract 1 if S>0 for tracks 1-
17
3733 'If S > 0 Then S -= 1 'Wrap around: Subtract 1 if S>0 for all tracks
3734 End If
3735 If TabStartS(T) = 255 Then
3736 'MsgBox(T.ToString + ":" + S.ToString)
3737 TabStartS(T) = S
3738 End If
3739 If SCnt < SMax Then GoTo NextSector
3740 Else
3741 S += 1
3742 If S >= SMax Then
3743 S = 0
3744 End If
3745 If SCnt < SMax Then GoTo NextSector
3746 End If
3747 Next
3748
3749 'IO.File.WriteAllBytes(UserFolder + "\OneDrive\C64\Coding\TabT.bin", TabT)
3750 'IO.File.WriteAllBytes(UserFolder + "\OneDrive\C64\Coding\TabS.bin", TabS)
3751 'IO.File.WriteAllBytes(UserFolder + "\OneDrive\C64\Coding\TabStartS.bin", TabStartS)
3752 'IO.File.WriteAllBytes(UserFolder + "\OneDrive\C64\Coding\TabSCnt.bin", TabSCnt)
3753
3754 Exit Sub
3755 Err:
3756 ErrCode = Err.Number
3757 MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3758
3759 End Sub
3760
3761 Public Sub GetILfromDisk()

```



```

3762     If DoOnErr Then On Error GoTo Err
3763
3764     IL0 = 256 - EORtransform(If(Disk(Track(18) + (0 * 256) + 250) <> 0, Disk(Track(18) + (0 * 256)
+ 250), EORtransform(4)))
3765     IL1 = 256 - EORtransform(If(Disk(Track(18) + (0 * 256) + 252) <> 0, Disk(Track(18) + (0 * 256)
+ 252), EORtransform(3)))
3766     IL2 = 256 - EORtransform(If(Disk(Track(18) + (0 * 256) + 253) <> 0, Disk(Track(18) + (0 * 256)
+ 253), EORtransform(3)))
3767     IL3 = 256 - EORtransform(If(Disk(Track(18) + (0 * 256) + 254) <> 0, Disk(Track(18) + (0 * 256)
+ 254), EORtransform(3)))
3768
3769     Exit Sub
3770 Err:
3771     ErrCode = Err.Number
3772     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3773
3774     End Sub
3775
3776     Private Sub UpdateBlocksFree()
3777         If DoOnErr Then On Error GoTo Err
3778
3779         If TracksPerDisk = ExtTracksPerDisk Then
3780             Dim ExtBlocksFree As Byte = If(BlocksFree > ExtSectorsPerDisk - StdSectorsPerDisk,
ExtSectorsPerDisk - StdSectorsPerDisk - BlocksUsedBySaver, BlocksFree)
3781             Disk(Track(18) + 4) += ExtBlocksFree
3782         End If
3783
3784         Exit Sub
3785 Err:
3786     ErrCode = Err.Number
3787     MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name
+ " Error")
3788     End Sub
3789
3790     'Public Function IsHexString(S As String) As Boolean
3791     'If DoOnErr Then On Error GoTo Err
3792     '
3793     'IsHexString = True
3794
3795     'For I As Integer = 1 To S.Length
3796     'Select Case Mid(LCase(S), I, 1)
3797     'Case " ", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "a", "b", "c", "d", "e", "f"
3798     'Case Else
3799     'IsHexString = False
3800     'Exit For
3801     'End Select
3802     'Next
3803
3804     'Exit Function
3805     'Err:
3806     'ErrCode = Err.Number
3807     'MsgBox(ErrorToString(), vbOKOnly + vbExclamation, Reflection.MethodBase.GetCurrentMethod.Name + "
Error")
3808
3809     'End Function
3810
3811 End Module
3812

```