```
Function calcChecksum_2byteXOR(hexEncodedData As String) As String
' Returns a 2 byte XOR checksum for error detection
' Does this by chain XOR 2 byte chunks of the input string
' The input string is expected to be hex encoded, and divisible by 4

If (Len(hexEncodedData) Mod 4) Then
' Try a 1 byte XOR
calcChecksum_2byteXOR = calcChecksum_1byteXOR(hexEncodedData)
Exit Function
End If

' Get the first bytes
Dim firstByteHex As String
firstByteHex = Mid(hexEncodedData, 1, 4)

' Only 1 byte long?
If Len(hexEncodedData) = 4 Then
calcChecksum_2byteXOR = firstByteHex
Exit Function
End If

Dim currentXor
currentXor = WorksheetFunction.Hex2Dec(firstByteHex)

For Index = 5 To Len(hexEncodedData) - 3 Step 4
' Get my bytes
Dim bytesInHex As String
bytesInHex = Mid(hexEncodedData, Index, 4)

' Convert to decimal
Dim bytesInDec
bytesInDec = WorksheetFunction.Hex2Dec(bytesInHex)

' XOR it
currentXor = currentXor Xor bytesInDec
Next

calcChecksum_2byteXOR = WorksheetFunction.Dec2Hex(currentXor, 4)

End Function

Function calcChecksum_1byteXOR(hexEncodedData As String) As String
' Returns a 1 byte XOR checksum for BASIC error detection
' Does this by chain XOR 1 byte chunks of the input string
' The input string is expected to be hex encoded, and divisible by 2

If (Len(hexEncodedData) Mod 2) Then
calcChecksum_1byteXOR = "FAIL"
Exit Function
End If

' Get the first bytes
Dim firstByteHex As String
firstByteHex = Mid(hexEncodedData, 1, 2)

' Only 1 byte long?
If Len(hexEncodedData) = 2 Then
calcChecksum_1byteXOR = firstByteHex
Exit Function
End If

Dim currentXor
currentXor = WorksheetFunction.Hex2Dec(firstByteHex)

For Index = 3 To Len(hexEncodedData) - 1 Step 2
' Get my bytes
```

```
Dim bytesInHex As String
bytesInHex = Mid(hexEncodedData, Index, 2)

' Convert to decimal
Dim bytesInDec
bytesInDec = WorksheetFunction.Hex2Dec(bytesInHex)

' XOR it
currentXor = currentXor Xor bytesInDec
Next

calcChecksum_1byteXOR = WorksheetFunction.Dec2Hex(currentXor, 2)

End Function

Function checkIntegrity(sheetname As String) As Boolean
' This module checks the checksum data of each line and the chain checksum to verify integrity
' Bad checksums are highlighted in red
' If all the row checksums are good and the final checksum is still red..  That's bad
' Returns true if all good
checkIntegrity = True

Dim sh As Worksheet
Set sh = Worksheets(sheetname)

sh.Cells.Interior.Color = vbWhite

Dim chainChecksum As Long
chainChecksum = 0

For Each rw In sh.Rows
' Turn blue for tracking purposes
sh.Cells(rw.Row, 2).Interior.Color = vbBlue
' First check to see if this row has line data, if not, then it only has the chainChecksum
If sh.Cells(rw.Row, 2).Value = "" Then
' Get the chainChecksum
Dim finalChecksum As Long
finalChecksum = WorksheetFunction.Hex2Dec(FixProblemCharacters(sh.Cells(rw.Row, 1).Value))

If (chainChecksum Xor finalChecksum) Then
' X Xor X = 0, if not 0 then there's an error
' Highlight the checksum cell red
sh.Cells(rw.Row, 1).Interior.Color = vbRed
checkIntegrity = False
Debug.Print "This data is corrupt"
Else
Debug.Print "This data is good to decode!"
End If

Exit For
End If

' Get my current row of data
Dim lineTgtChecksumHex As String
Dim currentData As String
lineTgtChecksumHex = FixProblemCharacters(sh.Cells(rw.Row, 1).Value)
currentData = FixProblemCharacters(sh.Cells(rw.Row, 2).Value)

' Calc the checksum
Dim lineActualChecksumHex As String
lineActualChecksumHex = calcChecksum_2byteXOR(currentData)

If lineActualChecksumHex = "FAIL" Then
sh.Cells(rw.Row, 1).Interior.Color = vbRed
checkIntegrity = False
Else
```

```
Dim tgtChecksumDec As Long
Dim actualChecksumDec As Long
tgtChecksumDec = WorksheetFunction.Hex2Dec(lineTgtChecksumHex)
actualChecksumDec = WorksheetFunction.Hex2Dec(lineActualChecksumHex)

' Test
If (tgtChecksumDec Xor actualChecksumDec) Then
' Should have been 0, it wasn't
' Highlight the checksum cell red
sh.Cells(rw.Row, 1).Interior.Color = vbRed
checkIntegrity = False
End If

' Chain the checksum
chainChecksum = chainChecksum Xor WorksheetFunction.Hex2Dec(lineActualChecksumHex)

End If
Next


End Function

Function ReplaceProblemCharacters(instring As String) As String
' Some characters will get confused in OCR, this is supposed to help
' B's 8's is a real problem
Dim fixedstring As String
fixedstring = Replace(instring, "B", "#")

' next is 1 to l (lowercase L) and 5 to S, but these are not valid hex characters, so easy to
' clean after

' D to Zero or Oh is the next biggest problem
fixedstring = Replace(fixedstring, "D", "?")

ReplaceProblemCharacters = fixedstring
' That may fix the majority of errors
End Function

Function FixProblemCharacters(instring As String) As String
' Reverses OCR problems
Dim fixedstring As String

fixedstring = Replace(instring, "#", "B")
fixedstring = Replace(fixedstring, "l", "1")
' Some "S" are supposed to be 8s.  No way to know ahead of time, need to use visual scan
fixedstring = Replace(fixedstring, "S", "5")
fixedstring = Replace(fixedstring, "G", "6")
fixedstring = Replace(fixedstring, "q", "6")
fixedstring = Replace(fixedstring, "b", "6")
fixedstring = Replace(fixedstring, "(", "C")
fixedstring = Replace(fixedstring, "?", "D")
' OCR likes to add spaces at the start
fixedstring = Replace(fixedstring, " ", "")
fixedstring = Replace(fixedstring, "~", "7")
' Dots on the paper become periods to OCR
fixedstring = Replace(fixedstring, ".", "")

FixProblemCharacters = fixedstring
' Well?
End Function


' Encode a file
' Opens the file and encodes each line, stored in column 2
' Calculates the checksum of each line and stores it in column 1
' Calculates the total checksum of each line, stored in column 1, after the final line
```

```
Sub EncodeFile()
' Declare needed inputs
Dim filename As String
Dim targetSheet As String
Dim maxBytesPerLine As Long

' Initialize needed inputs
filename = "C:\Users\1080119360A\Documents\Personal\work\bhat\sload.zip"
targetSheet = "sideload"

' maxBytesPerLine should be divisible by 4 to take advantage of the checksum function
' Consolas font in 8pt has about 130 chars avail per line, with space for the checksum value
maxBytesPerLine = 56

' Clear worksheet first
Dim sh As Worksheet
Set sh = Worksheets(targetSheet)
sh.Cells.Clear

' Open the file (no error catching here, it's a POC)
Open filename For Binary Access Read As #1
Dim fileLength As Long
fileLength = FileLen(filename)

' Initialize control variables
Dim currentLineByteCount As Integer
currentLineByteCount = 0
Dim currentRow As Long
currentRow = 2
Dim hexEncoded As String
hexEncoded = ""
Dim chainChecksum As Long
chainChecksum = 0

' Place file name in the sheet
sh.Cells(1, 1) = filename

' Encode the data
For currentFilePosn = 1 To fileLength
' Get a Byte
Dim thisByte As Byte
Get #1, , thisByte

' Encode it
Dim thisByteEncoded As String
thisByteEncoded = WorksheetFunction.Dec2Hex(thisByte, 2)

' Add to the string
hexEncoded = hexEncoded & thisByteEncoded

' Debug.Print hexEncoded

' Increment the byte count
currentLineByteCount = currentLineByteCount + 1

' Are we done with this line?
If currentLineByteCount >= maxBytesPerLine Then
' Store in the worksheet
sh.Cells(currentRow, 2).NumberFormat = "@"
sh.Cells(currentRow, 2) = ReplaceProblemCharacters(hexEncoded)

' Get the checksum
Dim lineChecksum As String
lineChecksum = calcChecksum_2byteXOR(hexEncoded)
```

```
' Store it
sh.Cells(currentRow, 1).NumberFormat = "@"
sh.Cells(currentRow, 1) = ReplaceProblemCharacters(lineChecksum)

' Chain the checksum
chainChecksum = chainChecksum Xor WorksheetFunction.Hex2Dec(lineChecksum)

' Increment the row count
currentRow = currentRow + 1

' Reset hexencoded
hexEncoded = ""

' Reset the byte count
currentLineByteCount = 0

End If
Next

' See if there is any residual data
If currentLineByteCount > 0 Then
' Need to save the last row
sh.Cells(currentRow, 2).NumberFormat = "@"
sh.Cells(currentRow, 2) = ReplaceProblemCharacters(hexEncoded)

' Get the checksum
lineChecksum = calcChecksum_2byteXOR(hexEncoded)

' Store it
sh.Cells(currentRow, 1).NumberFormat = "@"
sh.Cells(currentRow, 1) = ReplaceProblemCharacters(lineChecksum)

' Chain the checksum
chainChecksum = chainChecksum Xor WorksheetFunction.Hex2Dec(lineChecksum)

currentRow = currentRow + 1
End If

' Store the final checksum
sh.Cells(currentRow, 1).NumberFormat = "@"
sh.Cells(currentRow, 1).Value = ReplaceProblemCharacters(WorksheetFunction.Dec2Hex(chainChecksum, 4))

Debug.Print "Done."

Close #1

End Sub

Sub WriteToFile()
' Data is in the "Data Source" worksheet
' Open file in binary, write
' Iterate over every row, decode bytes in step 2, write to file
' done..

Dim dataSheetName As String
Dim filename As String

dataSheetName = "newdll-scan"

' First test data integrity
If Not checkIntegrity(dataSheetName) Then
Debug.Print "Cannot decode data, it is corrupt"
Exit Sub
End If

filename = "C:\Users\1080119360A\Documents\Personal\work\bhat\newdll-recovered.zip"
```

```
Dim rw As Range
Dim sh As Worksheet

Set sh = Worksheets(dataSheetName)

Open filename For Binary As #1

For Each rw In sh.Rows
' First check to see if this row has data
If sh.Cells(rw.Row, 2).Value = "" Then Exit For

' Get my current row of data
Dim currentData As String
currentData = FixProblemCharacters(sh.Cells(rw.Row, 2).Value)

For Index = 1 To Len(currentData) - 1 Step 2
' Get my byte
Dim currentByte As String
currentByte = Mid(currentData, Index, 2)

' Convert
Dim dataToWrite As Byte
dataToWrite = WorksheetFunction.Hex2Dec(currentByte)

' Write
Put #1, , dataToWrite
Next
Next

Close #1

Debug.Print "Write complete"

End Sub
```