:Cue:C.A.T.™ - Decoding the scanned string

This document was last updated November 8, 2000. Recent additions include more accurate decoding of :Cue:Cat "Cues" and a decoder written in Java.

I received several :Cue:Cat™ barcode readers as part of DigitalConvergence.com's promotion (they were free and given to anyone who asked for one at several Radio Shack stores locally).

They came with a CD with software that would provide "The simple way to surf the web". It was necessary to put their gadget (they called it a "cat", it looks very much like a mouse) between my keyboard and the keyboard jack on my computer. I was afraid that their software might not only "take me to the right web site" but tell who I was or where else I had been. I also did not like the idea of anybody's "free" software between my keyboard and my operating system. Therefore, I did not install their software, and began trying to simply use the new free (toy) barcode reader.

The problem was that the keystrokes generated did not look much like the barcodes I scanned as a test. Their brochure includes a UPC (Universal Product Code) sample that claims to be "6 91839 00001 1" (that is type 6, manufacturer # 91839, product # 00001, checkdigit 1). the scan came out (with the terminating CR/LF shown as ¶) as:

.C3nZC3nZC3nYCxD1C3D6CxnX.fHmc.DxPYE3b6C3nZC3jY.¶

The challenge was on! After several hours and dozens of other sample scans of everything I could find around the house with a barcode, I believe that the scan breaks down as:

U.P.C. 1 91839 00001 1
.C3nZC3nZC3nYCxD1C3D6CxnX.fHmc.DxPYE3b6C3nZC3jY.¶
Identifies a :Cue:C.A.T.|type|Actual scan data
(the periods are required punctuation)

For each of the sections other than the "Actual scan data", the scheme is similar to the "base64" encoding used for MIME (Multipurpose Internet Mail Extensions) encoding. They have used a slightly modified character table and then use an exclusive-or to (mildly) encrypt the data. For other scans than the special :Cue:Cat "Cues" (which are their "unique codes"), the XOR is with decimal 67 (hex. 43) and then simply use the result as an ASCII character. For the special :Cue:Cat "Cues", the XOR is with decimal 99 (hex. 63) and then interpret the result as 6 separate bytes of numeric data (but each with a range of 00 through 95 only -- values greater than 95 should be reduced by 64 (hex. 40)). Also, for :Cue:Cat "Cues", the result is prefaced by a capital "C" and another 2-digit decimal number equal to the value of the ASCII character after "CC" in the translated "type" string minus 32 (an ASCII blank).

The two :Cue:Cat units I got each provided a different but consistent identifier for the :Cue:Cat itself, that is, it appeared to be the "serial number" of the :Cue:Cat. I would guess from this that their database would be able to say to a client (manufacturer whose UPC you just scanned, perhaps) that the same user who just scanned your cereal box also reads

books by the following authors, magazines such as ***, scans packages of ***, etc. If their software install demands your name, address, E-mail address, etc., that data may also may be reported - no wonder they give these scanners away free.

An interesting sidelight: If your keyboard is set to "Caps Lock", the incoming scan will reverse the case of all the alphabetic characters. Assuming the serial numbers do not go into the bazillions, this is easily detected by the case of the first character of the serial number field. If that is lower case, simply reverse the case of all of the characters before decoding.

The values I found, to date, for the "type" are as follows:

fHmc	**UPA**	12 digits	U.P.C. barcodes on many products (Code UPC-A)
fGjX	**UA2**	14 digits	UPC + 2 digits such as "Bipad" numbers - identify a magazine or periodical and its edition number (e.g., month)
fGj2	**UA5**	17 digits	UPC + 5 digits
CNf7	**128**	20 digits	Many uses - allows alphanumeric characters
cGen	**IBN**	13 digits	EAN 13 such as ISBN coding on books
cHCf	**ITF**	(varies)	Interleaved 2-of-5 (many uses)
aabI	**CC!**		:Cue:Cat ":Cue" -- the coding for these is different: ("C 01" followed by six 2-digit numbers from 00 to 95) See the above discussion - any code "CC" is a ":Cue" with the first 2-digit number equal to the value of the 3rd character minus an ASCII blank (decimal 32).
ahb6	**C39**	(varies)	Code 3-of-9 -- includes alphanumerics
bNf7	**E28**	(varies)	
eW8q	**PLS**		Plessey
aaer	**CBR**		Codabar
dHak	**MSI**		MSI scanner

For my convenience in using my new, free barcode scanner, I wrote a Microsoft Access© 97 database which takes the scanned data and tries to translate it to a numeric value. You may download a copy at http://www.schnee.com/CatDecoder.mdb (it uses an autoexec macro to automatically open the (only) form in the database; if you want to look at the code before you run it, hold down the left shift key when loading the .mdb into Access© 97 or Access© 2000). No guarantees, here - use it if you like, fix it if you find something wrong, let me know if you make a significant improvement. This data and software are worth exactly what you are paying for them -- nothing. In order to use this database, you will need a licensed copy of Access©. If you don't have one but can read VBA code, the logic I used is shown below. The event procedure CatInput_AfterUpdate is invoked after the scan (which includes the terminating CR/LF).

If you would prefer, I have a command-line version of the decoder written in (rather simple) C. The source code may be downloaded at http://www.schnee.com/CatDecoder.c and a compiled version suitable for Intel-based DOS or Windows platforms may be downloaded at http://www.schnee.com/CatDecoder.exe.

:Cue:C.A.T.™ - Decoding the scanned string

I also wrote a Java class which implements the decoder. It is written much like a Java "bean" in that you instantiate the class once and then you may give it strings to decode, one at a time. It returns a boolean for whether the string was a valid scan. If the string was valid, you may retrieve the (decoded) serial number, the type and the scan value using "get" methods. That code (the source and the compiled class) as well as a test program that provides a simple command-line access to it and the class documentation (the result of using javadoc on the source with its embedded comments) are all available for download. See the page at http://www.schnee.com/cuecat.htm.

Cheers.

Dave Schnee E-mail to: dave@schnee.com

CatDecoder.mdb (VBA code for Microsoft Access© 97)

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Option Compare Binary
Option Explicit
Dim Srlstr As String
Dim Typstr As String
Dim Resstr As String
Dim DoingResult As Boolean
Dim IsCatCue As Boolean
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private Sub CatInput_AfterUpdate()

CatType = ""
Serial = ""
Result = ""

CatInput = Trim(CatInput)

' Validate format as ".Serialstring.Type.ScanValue."
If (Len(CatInput) < 34) Then GoTo Invalid
If Left(CatInput, 1) <> "." Then GoTo Invalid
If Mid(CatInput, 26, 1) <> "." Then GoTo Invalid
If Mid(CatInput, 31, 1) <> "." Then GoTo Invalid
If (Mid(CatInput, Len(CatInput), 1)) <> "." Then GoTo Invalid

' If first character of serial string is lowercase, invert case of whole string
'    because it is probably the result of being in "Caps Lock" mode.
If ((Mid(CatInput, 2, 1) = "c") Or (Mid(CatInput, 2, 1) = "d") Or
    (Mid(CatInput, 2, 1) = "e")) Then CatInput = Invert_case(CatInput)

Beep
Srlstr = Mid(CatInput, 2, 24)
Typstr = Mid(CatInput, 27, 4)
Resstr = Mid(CatInput, 32, Len(CatInput) - 32)

DoingResult = False
```

```
IsCatCue = False

Serial = decode(Srlstr)
CatType = decodetyp(Typstr)

DoingResult = True
Result = decode(Resstr)

Exit Sub
Invalid:
CatType = "Invalid"

End Sub
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private Function decode(coded As String) As String

Dim base64x As String
base64x = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+-"

Dim myc As String
Dim my4 As String
Dim LT As Long
Dim LI As Long

myc = coded
decode = ""

If (DoingResult And IsCatCue) Then
    decode = "C "
    LI = Asc(Mid(Typstr, 4, 1)) - 32
    decode = decode + Format(LI, "00")
    End If

Again:

my4 = Mid(myc, 1, 4)

LT = 0
If Len(my4) > 1 Then
    LI = InStr(1, base64x, Mid(my4, 1, 1), 0) - 1
    If (LI >= 0) Then LT = LT + (LI * 64 * 64 * 64)
    LI = InStr(1, base64x, Mid(my4, 2, 1), 0) - 1
    If (LI >= 0) Then LT = LT + (LI * 64 * 64)
    End If

If Len(my4) > 2 Then
    LI = InStr(1, base64x, Mid(my4, 3, 1), 0) - 1
    If (LI >= 0) Then LT = LT + (LI * 64)
    End If

If Len(my4) > 3 Then
    LI = InStr(1, base64x, Mid(my4, 4, 1), 0) - 1
    If (LI >= 0) Then LT = LT + LI
    End If

If (DoingResult And IsCatCue) Then
```

```
    If Len(my4) > 1 Then
        LI = (LT / 65536) Xor 99
        While (LI > 95)
            LI = LI - 64
            Wend
        decode = decode + " " + Format(LI, "00")
        End If

    If Len(my4) > 2 Then
        LI = ((LT / 256) And 255) Xor 99
                While (LI > 95)
            LI = LI - 64
            Wend
        decode = decode + " " + Format(LI, "00")
        End If

    If Len(my4) > 3 Then
        LI = (LT And 255) Xor 99
        While (LI > 95)
            LI = LI - 64
            Wend
        decode = decode + " " + Format(LI, "00")
        End If

Else

    If Len(my4) > 1 Then
        LI = (LT / 65536) Xor 67
        decode = decode + Chr$(LI)
        End If

    If Len(my4) > 2 Then
        LI = ((LT / 256) And 255) Xor 67
        decode = decode + Chr$(LI)
        End If

    If Len(my4) > 3 Then
        LI = (LT And 255) Xor 67
        decode = decode + Chr$(LI)
        End If

End If

If Len(myc) > 4 Then
    myc = Mid(myc, 5, 256)
    GoTo Again
    End If

End Function
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Dim TS As String

TS = decode(coded)
decodetyp = "'" + TS + "'"

If (TS = "UPA") Then decodetyp = "'" + TS + "' = " + "UPC-A (Universal Product
Code - type A)"
```

:Cue:C.A.T.™ - Decoding the scanned string

```
If (Mid(TS, 1, 2) = "CC") Then
    decodetyp = "'" + TS + "' = " + ":Cue:Cat :Cue"
    IsCatCue = True
    End If

If (Mid(TS, 1, 2) = "UA") Then
    decodetyp = "'" + TS + "' = " + "UPC + " + Mid(TS, 3, 1) + " digit(s)"
    End If

If (TS = "IBN") Then
    decodetyp = "'" + TS + "' = " + "EAN 13, such as I.S.B.N (Book number)"
    End If

If (TS = "ITF") Then decodetyp = "'" + TS + "' = " + "Interleaved 2-of-5"

If (TS = "C39") Then decodetyp = "'" + TS + "' = " + "Code 39"

If (TS = "PLS") Then decodetyp = "'" + TS + "' = " + "Plessey"

If (TS = "CBR") Then decodetyp = "'" + TS + "' = " + "Codabar"

End Function
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private Function Invert_case(inst As String) As String
Dim i As Integer
Dim i_case As String

i_case = ""

For i = 1 To Len(inst)
If (Mid(inst, i, 1) >= "a") And (Mid(inst, i, 1) <= "z") Then
    i_case = i_case + UCase(Mid(inst, i, 1))
Else
    If (Mid(inst, i, 1) >= "A") And (Mid(inst, i, 1) <= "Z") Then
    i_case = i_case + LCase(Mid(inst, i, 1))
    Else
        i_case = i_case + Mid(inst, i, 1)
        End If
    End If
Next i

Invert_case = i_case

End Function
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```