

Re: How to convert extra long strings into their equivalent Hex Strings in VBA (Word 2K)

Source:

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2008-05/msg00909.html>

- *From:* "Karl E. Peterson" <karl@xxxxxxxx>
 - *Date:* Wed, 14 May 2008 14:14:56 -0700
-

Hi Neil --

I have an interesting problem. I am trying to write code in VBA (Word 2K) wherein I have to convert a string-representation of really large numbers (upto 18 digits max) into its equivalent Hex String representation

e.g. `ConvertNumberToHex("999999999999999999") => "DE0B6B3A763FFFF"`

I have no problems as long as the number being passed to the function is less than the max allowed by the Long Type in VBA (Word 2K) viz. `&h7FFF FFFF = 2147483647`. But, when the input is larger than this number, I get overflows.

Trying the `Hex()` or `Val()` also results in overflows when the input number exceeds `&h7FFF FFFF`

Any ideas how to perform this function easily in VBA (Word 2K)?

Save the following code into its own BAS module, then call the `ConvertBase` function as needed. Full attribution, some documentation, and instructions for adding new bases are all in the comments. (Code indented to highlight wordwrap inherent in newsgroup posts.)

```
'-----  
' Modified by Karl E. Peterson, http://vb.mvps.org, on  
' 20-Nov-2007, to support Dell Service Tags and their  
' conversion to Express Service Code numbers.  
' All changes marked with 'KEP'  
'-----  
  
'-----  
' Module: mBases  
' (C) 2000 Trinet Ltd, http://www.trinet.co.uk  
' Author: R. Deeming (richard@xxxxxxxxxxxxxx)
```

Re: How to convert extra long strings into their equivalent Hex Strings in VBA (Word 2K)

' Purpose: To provide simple conversion between different
' number bases, including fractional parts.

' An example of advanced conversion is included:
' the sexagesimal (base 60) system. For this
' example, the numbers are represented as a
' 2-digit number from 0 to 59, followed by a colon.
' For example, ConvertBase("12:30:", ebSexagesimal)
' will return 750 (12 * 60 + 30). Note that the
' structure of the number is strict – "12:30" and
' "12:5:" are not valid numbers, but "12:30:" and
' "12:05:" are.

' When there are errors in the number, the code
' will raise error 13, with a meaningful message.
' To change the error number, change the constant
' ERROR_NUMBER, or modify the code.

' You may use and distribute this code, but you may not charge
' for it or present it as your own work. If you find any bugs
' in this code, please notify the author.
' This code is provided "As-Is" – if it doesn't work, we accept
' no responsibility.

' A complete explanation of bases is beyond the scope of this
' code, but a brief explanation is included.

' A number is represented as a series of digits, arranged in
' columns. The number of available digits is the same as the
' base of the number. Each column is indexed, starting at 0 for
' the "units" column, and the value of the column is equal to
' the base raised to the power of the index. We are accustomed to
' seeing this in decimal (base 10):

' Index 2 1 0 . -1 -2

' Value 2 3 4 . 2 5

' representing two hundred and thirty four and one quarter. This
' also applies to other bases. For example, in binary (base 2):

' Index 2 1 0 . -1 -2

' Value 1 0 1 . 0 1

' represents five and one quarter –
' $1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2}$
' = $4 + 0 + 1 + 0 + 0.25 = 5.25$

' This concept extends to bases of any whole number greater than
' 1, and is limited only by the number of available symbols.

' To add more bases, perform the following steps:
' 1. Add the base to the Bases enum, and set its value to
' the base (e.g. for base 4, add "ebBase4 = 4&"
'
' 2. If the base is greater than 10, or requires non-standard
' representation of the numbers, add a private const with
' the characters used to represent the digits (see
' hexadecimal for an example), or modify the DigitLength
' function and other functions (see ebSexagesimal)
'
' 3. In the procedures ConvertDigit and DeconvertDigit, add
' the new base to the select statement. If you are using a
' standard representation of a base <= 10, add the new enum
' value to the first select statement. Otherwise, copy and
' modify the hexadecimal option.

' Example:
' Add ebAlphabet, a base 26 option in which all digits are
' represented as letters of the alphabet.

' Private Const AlphaChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

' Public Enum Bases

' ...
' ebAlphabet = 26&

' Private Function ConvertDigit(...)

' ...
' Case ebAlphabet:
' ConvertDigit = Mid\$(AlphaChars, lngDigit + 1, 1)

' Private Function DeconvertDigit(...)

' ...
' Case ebAlphabet:
' lngTemp = InStr(1, AlphaChars, strDigit)
' If lngTemp = 0 Then
' err.raise error_number, "DeconvertDigit",
' "Invalid Alpha Character"
' Else
' DeconvertDigit = lngTemp - 1
' End If

Option Explicit

Re: How to convert extra long strings into their equivalent Hex Strings in VBA (Word 2K)

```
Private Const ERROR_NUMBER = 13&
```

```
Private Const HexChars = "0123456789ABCDEF"
```

```
Private Const AlphaChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
Private Const B36Chars = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ" & "KEP"
```

```
Public Enum Bases
```

```
    ebBinary = 2&
```

```
    ebOctal = 8&
```

```
    ebDecimal = 10&
```

```
    ebHexadecimal = 16&
```

```
    ebAlphabet = 26&
```

```
    ebHexatridecimal = 36& 'KEP: Added to support Dell Service Tags
```

```
    ebSexagesimal = 60& 'Base 60, e.g. time
```

```
End Enum
```

```
Public Function ExpressServiceCode(ByVal ServiceTag As String) As String
```

```
' Added by Karl E. Peterson, 20-Nov-2007, based on information found:
```

```
' http://www.creativyst.com/Doc/Articles/HT/Dell/DellNumb.htm
```

```
Dim i As Long
```

```
Dim sRet As String
```

```
Dim nDash As Long
```

```
' Check to make sure this is a valid Base36 number.
```

```
ServiceTag = Trim$(ServiceTag)
```

```
If Len(ServiceTag) Then
```

```
For i = 1 To Len(ServiceTag)
```

```
If InStr(B36Chars, Mid$(ServiceTag, i, 1)) = 0 Then
```

```
Exit Function
```

```
End If
```

```
Next i
```

```
' Decode from Base36 to decimal.
```

```
sRet = ConvertBase(ServiceTag, ebHexatridecimal)
```

```
' Add dashes after every three characters.
```

```
Do While nDash + 3 < Len(sRet)
```

```
sRet = Left$(sRet, nDash + 3) & "-" & Mid$(sRet, nDash + 4)
```

```
Debug.Print sRet
```

```
nDash = nDash + 4
```

```
Loop
```

```
End If
```

```
' Return results
```

```
ExpressServiceCode = sRet
```

```
End Function
```

```
Public Function ConvertBase(ByVal Number, ByVal FromBASE As Bases, _
```

```
Optional ByVal ToBASE As Bases = ebDecimal, Optional NumDecimals As Long = _
```

```
1, Optional Tolerance As Double = 1E-27, Optional PadTo As Long = 0) As _
```

```
Variant
```

```
'Convert a number from one base to another.
```

```
'Parameters:
```

Re: How to convert extra long strings into their equivalent Hex Strings in VBA (Word 2K)

```
' Number A numeric value (when FromBASE = ebDecimal) or
' a string representing the number to convert
,
' FromBASE The base to convert from (enumeration)
,
' ToBASE (Optional) The base to convert to. Default = Decimal
,
' NumDecimals (Optional) The number of decimal places to include
' when converting a fractional number to a non-decimal
' Specify 0 for integer only, or -1 to use tolerance.
' (This prevents problems with infinite loops)
,
' Tolerance (Optional) The value at which to terminate the
' fractional representation. If NumDecimals = -1
' and Tolerance = 0, no attempt will be made, to
' avoid an infinite loop. The sign of the tolerance
' is ignored.
,
' PadTo (Optional) Used to pad the non-decimal number
' to a given length. EG, binary numbers are
' normally shown with a multiple of 8 digits, so
' you would specify 8. Use 0 to avoid padding.
,
Returns:
' Either a double (if ToBASE = ebDecimal) or a string representing
' the converted number.
Dim dblDec As Double

If FromBASE = ebDecimal Then
If IsNumeric(Number) Then
dblDec = CDbI(Number)
Else
Err.Raise ERROR_NUMBER, "ConvertBase", "Not a decimal number"
End If
Else
dblDec = ConvertBase2Dec(CStr(Number), FromBASE)
End If
If ToBASE = ebDecimal Then
ConvertBase = dblDec
Else
ConvertBase = ConvertDec2Base(dblDec, ToBASE, NumDecimals, Tolerance,
PadTo)
End If
End Function

Public Function LogB(ByVal dblNumber As Double, ByVal lBase As Long) As Double
LogB = Log(dblNumber) / Log(lBase)
End Function

Private Function DigitLength(Base As Bases) As Long
'Return the length of a digit in a given base
```

Re: How to convert extra long strings into their equivalent Hex Strings in VBA (Word 2K)

```
Select Case Base
Case ebSexagesimal:
DigitLength = 3
```

```
'Add other special cases here
```

```
Case Else 'ebBinary, ebOctal, ebDecimal, ebHexadecimal
DigitLength = 1
End Select
End Function
```

```
Private Function Floor(ByVal Number As Double) As Double
'Return the floor of the number
'(the highest whole number less than or equal to the number)
If Int(Number) > Number Then
Floor = Int(Number) - 1
Else
Floor = Int(Number)
End If
End Function
```

```
Private Function GetNumDec(dblTemp As Double, PadTo As Long, Base As Bases) As
Long
Dim lTemp As Long, lTempPad As Double
'Return the number of digits required to represent
'the number dblTemp in the specified base, padded
'to the nearest multiple of PadTo
If dblTemp = 0 Then
lTemp = 1
Else
lTemp = Floor(Log(dblTemp) / Log(Base)) + 1
End If
If PadTo > 1 Then
lTempPad = lTemp / CDbI(PadTo)
If lTempPad > Floor(lTempPad) Then
lTempPad = 1 + Floor(lTempPad) - lTempPad
lTempPad = lTempPad * PadTo
lTemp = lTemp + lTempPad
End If
End If
GetNumDec = lTemp
End Function
```

```
Private Function ConvertDigit(lngDigit As Long, Base As Bases) As String
'Convert a single digit to the specified base
If lngDigit >= Base Then
Err.Raise ERROR_NUMBER, "ConvertDigit", "Invalid digit for base"
Else
Select Case Base
Case ebBinary, ebOctal, ebDecimal
ConvertDigit = CStr(lngDigit)
```

Re: How to convert extra long strings into their equivalent Hex Strings in VBA (Word 2K)

Case ebHexadecimal

```
ConvertDigit = Mid$(HexChars, lngDigit + 1, 1)
```

Case ebAlphabet

```
ConvertDigit = Mid$(AlphaChars, lngDigit + 1, 1)
```

Case ebHexatridecimal

'KEP: Added to support Dell Service Tags

```
ConvertDigit = Mid$(B36Chars, lngDigit + 1, 1)
```

Case ebSexagesimal

```
ConvertDigit = Right$("00" & CStr(lngDigit), 2) & ":"
```

Case Else

```
Err.Raise ERROR_NUMBER, "ConvertDigit", "Unknown base"
```

```
End Select
```

```
End If
```

```
End Function
```

Private Function DeconvertDigit(strDigit As String, Base As Bases) As Long

'Convert a single digit from the specified base to decimal

```
Dim lngTemp As Long
```

```
Select Case Base
```

```
Case ebBinary, ebOctal, ebDecimal
```

```
If IsNumeric(strDigit) Then
```

```
lngTemp = CLng(strDigit)
```

```
If lngTemp < Base Then
```

```
DeconvertDigit = lngTemp
```

```
Else
```

```
Err.Raise ERROR_NUMBER, "DeconvertDigit", "Invalid digit for base"
```

```
End If
```

```
Else
```

```
Err.Raise ERROR_NUMBER, "DeconvertDigit", "Invalid character"
```

```
End If
```

Case ebHexadecimal

```
lngTemp = InStr(1, HexChars, UCase$(strDigit))
```

```
If lngTemp = 0 Then
```

```
Err.Raise ERROR_NUMBER, "DeconvertDigit", "Invalid digit for base"
```

```
Else
```

```
DeconvertDigit = lngTemp - 1
```

```
End If
```

Case ebAlphabet

```
lngTemp = InStr(1, AlphaChars, UCase$(strDigit))
```

```
If lngTemp = 0 Then
```

```
Err.Raise ERROR_NUMBER, "DeconvertDigit", "Invalid Alpha Character"
```

```
Else
```

```
DeconvertDigit = lngTemp - 1
```

```
End If
```

```
Case ebHexatridecimal
'KEP: Added to support Dell Service Tags
lngTemp = InStr(1, B36Chars, UCase$(strDigit))
If lngTemp = 0 Then
Err.Raise ERROR_NUMBER, "DeconvertDigit", "Invalid digit for base"
Else
DeconvertDigit = lngTemp - 1
End If
```

```
Case ebSexagesimal
If Len(strDigit) = 3 Then
If Right$(strDigit, 1) = ":" And IsNumeric(Left$(strDigit, 2)) Then
lngTemp = CLng(Left$(strDigit, 2))
If lngTemp < Base Then
DeconvertDigit = lngTemp
Else
Err.Raise ERROR_NUMBER, "DeconvertDigit", "Invalid digit for
base"
End If
Else
Err.Raise ERROR_NUMBER, "DeconvertDigit", "Invalid digit for base"
End If
Else
Err.Raise ERROR_NUMBER, "DeconvertDigit", "Invalid digit for base"
End If
```

```
Case Else
Err.Raise ERROR_NUMBER, "DeconvertDigit", "Unknown base"
End Select
End Function
```

```
Private Function ConvertDec2Base(ByVal Number, ByVal Base As Bases, _
Optional NumDecimals As Long = -1, Optional Tolerance As Double = 1E-27, _
Optional PadTo As Long = 0) As String
'Convert Number from decimal to the specified base,
'with NumDecimals fractional digits (or to within tolerance),
'padded to the nearest multiple of PadTo
Dim dblTemp As Double
Dim lCDec As Long
Dim lDigit As Long
Dim dblPwr As Double
Dim strTemp As String
```

```
If Not IsNumeric(Number) Then
Err.Raise ERROR_NUMBER, "ConvertDec2Base", "Number must be decimal"
ElseIf Base < 2 Then
Err.Raise ERROR_NUMBER, "ConvertDec2Base", "Invalid base"
Else
'Negative tolerance could cause loops
Tolerance = Abs(Tolerance)
```

Re: How to convert extra long strings into their equivalent Hex Strings in VBA (Word 2K)

```
dblTemp = CDbI(Number)
If dblTemp < 0 Then
strTemp = "-"
dblTemp = -dblTemp
End If
```

```
ICDec = GetNumDec(dblTemp, PadTo, Base)
```

```
'Integer part
If ICDec = 0 Then
strTemp = strTemp & "0"
Else
Do Until ICDec = 0
ICDec = ICDec - 1
dblPwr = Base ^ ICDec
IDigit = 0
Do While dblTemp >= dblPwr
IDigit = IDigit + 1
dblTemp = dblTemp - dblPwr
Loop
strTemp = strTemp & ConvertDigit(IDigit, Base)
Loop
End If
```

```
'Fractional part
If dblTemp > Tolerance And (NumDecimals > 0 Or (NumDecimals = -1 And _
Tolerance > 0)) Then
strTemp = strTemp & "."
Do While dblTemp > Tolerance And (ICDec > (-NumDecimals) Or _
NumDecimals = -1)
ICDec = ICDec - 1
dblPwr = Base ^ ICDec
IDigit = 0
Do While dblTemp >= dblPwr
IDigit = IDigit + 1
dblTemp = dblTemp - dblPwr
Loop
strTemp = strTemp & ConvertDigit(IDigit, Base)
Loop
End If
ConvertDec2Base = strTemp
End If
End Function
```

```
Private Function ConvertBase2Dec(ByVal Number As String, ByVal Base As Bases) As
Double
Dim dblTemp As Double
Dim strDigit As String, lngDigit As Long, i As Long
Dim lngPwr As Long, lngSign As Long, lngDigitSize
```

Re: How to convert extra long strings into their equivalent Hex Strings in VBA (Word 2K)

```
'Convert the number from the specified base to decimal
If Base < 2 Then
Err.Raise ERROR_NUMBER, "ConvertBase2Dec", "Invalid Base"
Else
lngDigitSize = DigitLength(Base)
lngPwr = 0
lngSign = 1
i = 1
Do Until i > Len(Number)
strDigit = Mid$(Number, i, lngDigitSize)
If Left$(strDigit, 1) = "." Then
i = i + 1
If lngPwr = 0 Then
lngPwr = 1
Else
Err.Raise ERROR_NUMBER, "ConvertBase2Dec", "More than one decimal
point"
End If

ElseIf Left$(strDigit, 1) = "-" Then
i = i + 1
If lngPwr = 0 And dblTemp = 0 Then
lngSign = -lngSign
Else
Err.Raise ERROR_NUMBER, "ConvertBase2Dec", "Invalid negation"
End If

Else
i = i + lngDigitSize
lngDigit = DeconvertDigit(strDigit, Base)
dblTemp = dblTemp * Base + lngDigit
lngPwr = lngPwr * Base
End If
Loop

If lngPwr > 1 Then
ConvertBase2Dec = Cdbl(lngSign) * (dblTemp / Cdbl(lngPwr))
Else
ConvertBase2Dec = Cdbl(lngSign) * dblTemp
End If
End If
End Function
```

Enjoy... Karl

..NET: It's About Trust!

<http://vfred.mvps.org>

.