

## **Table of Contents**

**Introduction**

**Overview of the Routines**

**Thanks and Acknowledgments**

**Disclaimer**

**Comments & Suggestions**

## Introduction

One of the things that Microsoft left out of Visual Basic for Windows were the MKDMBF\$, MKSMBF\$, CVDMBF, and CVSMBF functions. These functions converted the old Microsoft Binary Format numbers used by BASICA or older versions of QuickBASIC (3.0 and earlier) to the new IEEE format. Of course, there never was an internal function in QuickBASIC to convert Turbo Pascal 6-byte Reals into numbers that QB could understand. Two functions included in this package, MDKTPR\$ and CVDTPR, allow you to perform this conversion. They were included in this package because these routines were included in the original C routines.

To assist Visual Basic programmers who need a way to access or convert the data created by these older programs these routines are presented for your use without any charge or registration required.

This is the second version of these routines to be released. The first version was put together rather quickly to meet my own and others deadlines. It was quick and dirty and was in need of some refinements and bug fixes. The most important change to the conversion routines was changing them from Subs to Functions for this version and getting rid of several global variables. The other major change was putting everything into this help file. All of the conversion routines are in this file, as well as in the CVDMBF.BAS file that you can add to your VB project. If you don't want to load all of the routines into your project you can use the **Copy Topic** button to just get the routines you want easily.

In case you are looking for routines to do the other MK?? and CV?? functions, they are included in this package as well, since they are required for the proper operation of the MBF routines. These routines are from a Microsoft Knowledge base article that described using the hmemcpy API call to perform this operation.

## **Overview of the Routines**

### MKDMBF\$ ( OldNumberDP as Double )

This routine converts a IEEE double precision number to a MBF 8 byte string.

### MKSMBF\$ ( OldNumberSP As Single )

This routine converts a IEEE single precision number to a MBF 4 byte string.

### MKDTPR\$ ( OldNumberDP as Double )

This routine converts a IEEE double precision number to a Turbo Pascal 6 byte string.

### CVDMBF ( OldStringDP As String ) as Double

This routine converts a MBF 8 byte string into a IEEE double precision number.

### CVSMBF ( OldStringSP as String ) as Single

This routine converts a MBF 4 byte string into a IEEE single precision number.

### CVDTPR ( OldStringTP as String ) as Double

This routine converts a Turbo Pascal 6 byte real string into a IEEE double precision number.

### Other MK??/CV?? Routines

```

Function MKDMBF$ (OldNumberDP As Double)
    Dim X, Sign, Exponent As Integer
    Dim NewNum As String
    Dim OldString As String
    Static ONA(0 To 7), NNA(0 To 7)
    OldNum# = OldNumberDP
    OldString = MKD$(OldNum#)
    For X = 0 To 7
        ONA(X) = Asc(Mid$(OldString, X + 1, 1))
    Next
    Sign = ONA(7) And 128
    Exponent = ((ONA(7) And 127) * 2 ^ 4 And 255) + (ONA(6) 2 ^ 4 And 255)
    If Exponent Then Exponent = (Exponent + 129 - 1023) And 255
    For X = 6 To 1 Step -1
        NNA(X) = ONA(X) * 2 ^ 4 And 255
        NNA(X) = NNA(X) Or ONA(X - 1) 2 ^ 4 And 255
    Next
    For X = 0 To 5
        NNA(X) = NNA(X) 2 ^ 1 And 255
        NNA(X) = NNA(X) Or NNA(X + 1) * 2 ^ 7 And 255
    Next
    NNA(6) = NNA(6) 2 ^ 1 And 255
    NNA(6) = NNA(6) Or Sign
    NNA(7) = Exponent
    MKDMBF$ = Space$(8)
    For X = 0 To 7
        Mid$(MKDMBF$, X + 1, 1) = Chr$(NNA(X))
    Next
End Function

```

```

Function MKSMBF$ (OldNumberSP As Single)
    Dim X, Sign, Exponent As Integer
    Dim OldString As String
    ReDim ONA(0 To 7)
    ReDim NNA(0 To 3)
    OldString = MKD$(CDBl(OldNumberSP))
    For X = 0 To 7
        ONA(X) = Asc(Mid$(OldString, X + 1, 1))
    Next
    Sign = ONA(7) And 128
    Exponent = ((ONA(7) And 127) * 2 ^ 4 And 255) + (ONA(6) 2 ^ 4 And 255)
    If Exponent Then Exponent = (Exponent + 129 - 1023) And 255
    For X = 2 To 0 Step -1
        NNA(X) = ONA(X + 4) * 2 ^ 4 And 255
        NNA(X) = NNA(X) Or ONA(X + 3) 2 ^ 4 And 255
    Next
    For X = 0 To 1
        NNA(X) = NNA(X) 2 ^ 1 And 255
        NNA(X) = NNA(X) Or NNA(X + 1) * 2 ^ 7 And 255
    Next
    NNA(2) = NNA(2) 2 ^ 1 And 255
    NNA(2) = NNA(2) Or Sign
    NNA(3) = Exponent
    MKSMBF$ = Space$(4)
    For X = 0 To 3
        Mid$(MKSMBF$, X + 1, 1) = Chr$(NNA(X))
    Next
End Function

```

```

Function MKDTPR$ (OldNumberDP As Double)
    Dim X, Sign, Exponent As Integer
    Dim NewNum, OldString As String
    Static ONA(0 To 7), NNA(0 To 5)
    OldNum# = OldNumberDP
    OldString = MKD$(OldNum#)
    For X = 0 To 7
        ONA(X) = Asc(Mid$(OldString, X + 1, 1))
    Next
    Sign = ONA(7) And 128
    Exponent = (((ONA(7) And 127) * 2 ^ 4 And 255) + (ONA(6) 2 ^ 4 And 255) + 129 - 1023) And 255
    For X = 5 To 1 Step -1
        NNA(X) = ONA(X + 1) * 2 ^ 4 And 255
        NNA(X) = NNA(X) Or ONA(X) 2 ^ 4 And 255
    Next
    For X = 1 To 4
        NNA(X) = NNA(X) 2 ^ 1 And 255
        NNA(X) = NNA(X) Or NNA(X + 1) * 2 ^ 7 And 255
    Next
    NNA(5) = NNA(5) 2 ^ 1 And 255
    NNA(5) = NNA(5) Or Sign
    NNA(0) = Exponent
    MKDTPR$ = Space$(6)
    For X = 0 To 5
        Mid$(MKDTPR$, X + 1, 1) = Chr$(NNA(X))
    Next
End Function

```

```

Function CVDMBF (OldStringDP As String) As Double
    Dim X, Sign, Exponent As Integer
    Dim NewNum As String
    Static ONA(0 To 7), NNA(0 To 7)
    For X = 0 To 7
        ONA(X) = Asc(Mid$(OldStringDP, X + 1, 1)): NNA(X) = 0
    Next
    Sign = ONA(6) And 128
    Exponent = ONA(7) - 129 + 1023
    NNA(6) = Exponent * 2 ^ 4 And 255
    NNA(7) = (Exponent 2 ^ 4 And 255) Or Sign
    For X = 6 To 1 Step -1
        ONA(X) = ONA(X) * 2 ^ 1 And 255
        ONA(X) = ONA(X) Or ONA(X - 1) 2 ^ 7 And 255
    Next
    ONA(0) = ONA(0) * 2 ^ 1 And 255
    For X = 6 To 2 Step -1
        NNA(X) = NNA(X) Or ONA(X) 2 ^ 4 And 255
        NNA(X - 1) = ONA(X) * 2 ^ 4 And 255
    Next
    For X = 0 To 7
        NewNum = NewNum + Chr$(NNA(X))
    Next
    CVDMBF = CVD(NewNum)
End Function

```

```

Function CVSMBF (OldStringSP As String) As Single
    Dim X, Sign, Exponent As Integer
    Dim NewNum As String
    Static ONA(0 To 3), NNA(0 To 7)
    For X = 0 To 3
        ONA(X) = Asc(Mid$(OldStringSP, X + 1, 1))
    Next
    For X = 0 To 7
        NNA(X) = 0
    Next
    Sign = ONA(2) And 128
    Exponent = ONA(3) - 129 + 1023
    NNA(6) = Exponent * 2 ^ 4 And 255
    NNA(7) = (Exponent * 2 ^ 4 And 255) Or Sign
    For X = 2 To 1 Step -1
        ONA(X) = ONA(X) * 2 ^ 1 And 255
        ONA(X) = ONA(X) Or ONA(X - 1) * 2 ^ 7 And 255
    Next
    ONA(0) = ONA(0) * 2 ^ 1 And 255
    For X = 6 To 4 Step -1
        NNA(X) = NNA(X) Or ONA(X - 4) * 2 ^ 4 And 255
        NNA(X - 1) = ONA(X - 4) * 2 ^ 4 And 255
    Next
    For X = 0 To 7
        NewNum = NewNum + Chr$(NNA(X))
    Next
    CVSMBF = CSng(CVD(NewNum))
End Function

```

```

Function CVDTPR (OldStringTP As String) As Double
    Dim X, Sign, Exponent As Integer
    Dim NewNum As String
    Static ONA(0 To 5), NNA(0 To 7)
    For X = 0 To 5
        ONA(X) = Asc(Mid$(OldStringTP, X + 1, 1))
    Next
    For X = 0 To 7
        NNA(X) = 0
    Next
    Sign = ONA(5) And 128
    Exponent = ONA(0) - 129 + 1023
    NNA(6) = Exponent * 2 ^ 4 And 255
    NNA(7) = (Exponent * 2 ^ 4 And 255) Or Sign
    For X = 5 To 2 Step -1
        ONA(X) = ONA(X) * 2 ^ 1 And 255
        ONA(X) = ONA(X) Or ONA(X - 1) * 2 ^ 7 And 255
    Next
    ONA(0) = ONA(0) * 2 ^ 1 And 255
    For X = 6 To 2 Step -1
        NNA(X) = NNA(X) Or ONA(X - 1) * 2 ^ 4 And 255
        NNA(X - 1) = ONA(X - 1) * 2 ^ 4 And 255
    Next
    For X = 0 To 7
        NewNum = NewNum + Chr$(NNA(X))
    Next
    CVDTPR = CVD(NewNum)
End Function

```

## **Other MK??/CV?? Routines**

MKI\$  
MKL\$  
MKS\$  
MKD\$

CVI  
CVL  
CVS  
CVD

```
Function MKI$ (X As Integer)
    temp$ = Space$(2)
    hmemcpy ByVal temp$, X, 2
    MKI$ = temp$
End Function
```

```
Function MKL$ (X As Long)
    temp$ = Space$(4)
    hmemcpy ByVal temp$, X, 4
    MKL$ = temp$
End Function
```

```
Function MKS$ (X As Single)
    temp$ = Space$(4)
    hmemcpy ByVal temp$, X, 4
    MKS$ = temp$
End Function
```

```
Function MKD$ (X As Double)
    temp$ = Space$(8)
    hmemcpy ByVal temp$, X, 8
    MKD$ = temp$
End Function
```

```
Function CVI (X As String) As Integer
    If Len(X) <> 2 Then
        MsgBox "Illegal Function Call"
        Stop
    End If
    hmemcpy temp#, ByVal X, 2
    CVI = temp#
End Function
```

```
Function CVL (X As String) As Long
    If Len(X) <> 4 Then
        MsgBox "Illegal Function Call"
        Stop
    End If
    hmemcpy temp#, ByVal X, 4
    CVL = temp#
End Function
```

```
Function CVS (X As String) As Single
    If Len(X) <> 4 Then
        MsgBox "Illegal Function Call"
        Stop
    End If
    hmemcpy temp#, ByVal X,4
    CVS = temp#
End Function
```

```
Function CVD (X As String) As Double
    If Len(X) <> 8 Then
        MsgBox "Illegal Function Call"
        Stop
    End If
    hmemcpy temp#, ByVal X, 8
    CVD = temp#
End Function
```

## **Declaration for and Description of the hmemcpy API Call**

The hmemcpy routine copies bytes from a source buffer to a destination buffer. This routine is used to copy the value of each byte in a numeric value, in the case of the MK??? functions, or string, in the case of the CV?? functions, to a corresponding byte in the destination buffer. The hmemcpy routine requires you to pass strings by value (ByVal) instead of passing the location of the string descriptor. It is also required that the string size is initialized to the correct number of characters.

This declaration should be put in the General|Declarations section of your program and should be placed all on one line. See the other MK??/CV?? routines for how this call is used.

**Declare Sub hmemcpy Lib "kernel" (hpdDest As Any, hpdSource As Any, ByVal cbCopy As Long)**

## Thanks and Acknowledgments

First I would like to thank **David W. Terry** for putting the original C routines on CompuServe for all to use. These original routines are in **REALCN.ZIP** in the IBMPRO forum C file library.

I would also like to thank **Bryan Donaldson**, **Fred Bunn**, and **Mark Howard**, all of whom contributed to the creation and debugging of these routines. Also, a thank you to **Kenneth Jamieson** who suggested several of the refinements found in this version.

## **Disclaimer**

No warranties or guarantees are made for these routines and you are solely responsible for their use or misuse. The routines are considered Public Domain. You may use or modify them as needed.

All the company names and products mentioned in this document are trademarks of the respective companies.

Since I don't have a copy of Turbo Pascal, I was unable to test the TP related routines as thoroughly as I would have liked. If you encounter problems with them, let me know so that I can make corrections to it.

## **Comments & Suggestions**

I would like to hear from anyone who gets any use out of these routines. I am also available for programming projects and consulting in Visual Basic. You may contact me at:

**J. Frank Carr  
1532 Amber Trail  
Duluth, GA 30136 USA**

**Voice Mail: (404) 880-5762**

**CIS: 75120, 2420**

**America OnLine: JFCarr**



