

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

Contents

Overview

Subroutine Index

Revision History

Conditions for Distribution

SoftCircuits Programming
P.O. Box 16262
Irvine, CA 92713
CompuServe: 72134,263

Redistributed by Permission.

This Help file has been created by Ian Carter, and is distributed with the permission of SoftCircuits Programming.
If you have any queries regarding this Help file, do NOT contact SoftCircuits Programming, but contact me by one of the following methods:-

Email: ian.carter@nildram.co.uk
CIS: 100435,3040

VB-ASM, Version 1.30

Copyright © 1994-95 SoftCircuits Programming

Overview

VB-ASM is a DLL that was written to help Visual Basic programmers accomplish tasks that are either difficult, or impossible to do in Visual Basic alone. The DLL contains a number of helpful routines and was written entirely in assembly language making it highly optimized. In addition, VB-ASM is free and you can use and distribute VB-ASM with your own programs as long as you follow the conditions outlined below.

To use VB-ASM in your Visual Basic program, you should copy the VBASM.TXT file to one of the .BAS modules in your project. This file contains the declarations for all of the VB-ASM subroutines and functions. You must also place the DLL itself where Windows can find it (normally in the Windows system directory). You can then call these routines as you would call any other DLL routine. See the Visual Basic documentation for additional information about calling DLL routines from Visual Basic.

WARNING: Visual Basic prevents you from making most errors that would adversely affect the system. When you use this or any other DLL, Visual Basic can no longer prevent these types of errors. Under Windows protected mode, most errors will result in a General Protection Fault (GPF). However, it is possible, using VB-ASM, to corrupt Windows, DOS or even the files on your disk. Use caution when working with any DLL and be sure to save and backup your files often.

This DLL was created to provide help to other Visual Basic programmers. If you find a problem or have a suggestion for making the DLL or associated documentation more helpful, please share your knowledge and let us know.

VB-ASM was written by Jonathan Wood. Thanks also to Douglas Marquardt for his input and Andrew Schulman (author of Undocumented DOS, Second Edition, Addison-Wesley) for kindly answering questions.

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

Conditions for Distribution

This program may be used and distributed freely on the condition that it is distributed in full and unchanged, and that no fee is charged for such use and distribution with the exception or reasonable media and shipping charges.

You may also incorporate any or all portions of this program, and/or include the VB-ASM DLL, as part of your own programs and distribute such programs without payment of royalties on the condition that such program do not duplicate the overall functionality of VB-ASM and/or any of its demo programs, and that you agree to the following disclaimer:

WARNING: Accessing the low-level services of Windows, DOS and the ROM-BIOS using VB-ASM is an extremely powerful technique that, if used incorrectly, can cause possible permanent damage and/or loss of data. You are responsible for determining appropriate use of any and all files included in this package. SoftCircuits will not be held liable for any damages resulting from the use of these files.

SOFTCIRCUITS SPECIFICALLY DISCLAIMS ALL WARRANTIES, INCLUDING, WITHOUT LIMITATION, ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS.

UNDER NO CIRCUMSTANCES WILL SOFTCIRCUITS BE LIABLE FOR SPECIAL, INCIDENTAL, CONSEQUENTIAL, INDIRECT, OR ANY OTHER DAMAGES OR CLAIMS ARISING FROM THE USE OF THIS PRODUCT, INCLUDING LOSS OF PROFITS OR ANY OTHER COMMERCIAL DAMAGES, EVEN IF WE HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Please contact SoftCircuits Programming if you have any questions concerning these conditions.

Subroutine Index

All the subroutines contained within the VB-ASM DLL in alphabetical order.

vbGetCtrlHwnd
vbGetCtrlModel
vbGetCtrlName
vbGetData
vbGetDriveType
vbGetLongPtr
vbHiByte
vbHiWord
vbInp
vbInpw
vbInterrupt
vbInterruptX
vbLoByte
vbLoWord
vbMakeLong
vbMakeWord
vbOut
vbOutw
vbPeek
vbPeekw
vbPoke
vbPokew
vbRealModelIntX
vbRecreateCtrl
vbSAdd
vbSetData
vbShiftLeft
vbShiftLeftLong
vbShiftRight
vbShiftRightLong
vbSSeg
vbVarPtr
vbVarSeg

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbGetCtrlHwnd

Declaration: Declare Function vbGetCtrlHwnd Lib "VBASM.DLL" (*Ctrl* As Any) As integer

Description: Returns the window handle for the control specified by *Ctrl*, or 0 if the control is a graphical control (graphical controls have no window associated with them).

VB-ASM, Version 1.30

Copyright © 1994-95 SoftCircuits Programming

vbGetCtrlModel

Declaration: Declare Function vbGetCtrlModel Lib "VBASM.DLL" (*Ctrl* As Any) As Long

Description: This function returns a long pointer to the MODEL structure of the Visual Basic form or control specified by *Ctrl*. The MODEL structure is used internally by all Visual Basic controls and is defined as follows:

```
Type MODEL
    usVersion As Integer           'VB version used by control
    fl As Long                    'Bitfield structure
    pctlproc As Long              'The control proc.
    fsClassStyle As Integer       'Window class style
    flWndStyle As Long            'Default window style
    cbCtlExtra As Integer         '# bytes alloc'd for HCTL structure
    idBmpPalette As Integer       'BITMAP id for tool palette
    npszDefCtlName As Integer     'Default control name prefix
    npszClassName As Integer     'Visual Basic class name
    npszParentClassName As Integer 'Parent window class if subclassed
    npproplist As Integer         'Property list
    npeventlist As Integer       'Event list
    nDefProp As String * 1        'Index of default property
    nDefEvent As String * 1       'Index of default event
    nValueProp As String * 1      'Index of control value property
    usCtlVersion As Integer       'Identifies the current version of
                                'the custom control. The values 1
                                'and 2 are reserved for custom
                                'controls created with VB 1.0
                                'and 2.0
End Type
```

IMPORTANT: In Visual Basic versions higher than 3.0, the MODEL structure is read-only.

See Also: [vbGetCtrlName](#), [vbRecreateCtrl](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbGetCtrlName

Declaration: Declare Function vbGetCtrlName Lib "VBASM.DLL" (*Ctrl* As Any) As String

Description: Returns the name property of the control or form specified by *Ctrl*. Normally, Visual Basic will not allow you to access a control's Name property at run-time.

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbGetData

Declaration: Declare Sub vbGetData Lib "VBASM.DLL" (ByVal *Pointer* As Long, *Variable* As Any, ByVal *nCount* As Integer)

Description: Copies the data from the memory location pointed to by *Pointer* to *Variable*. *nCount* specifies the number of bytes to be copied. This function is useful when you need access to data not within your program.

See Also: [vbSetData](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbGetDriveType

Declaration: Declare Function vbGetDriveType Lib "VBASM.DLL" (ByVal *DriveStr* As String) As Integer

Description: Returns a value that indicates the drive type for the drive letter specified by *DriveStr*. If *DriveStr* contains more than one character, only the first character is referenced.

This function works similar to the Windows API function GetDriveType except that vbGetDriveType also detects floppy, CD-ROM and RAM drives. The return value will be one of the following constants (which are defined in VBASM.TXT):

Constant	Value	Meaning
VBDRIVE_REMOVABLE	&H1002	Unknown removable media device such as a Bernoulli box.
VBDRIVE_FIXED	&H1003	Hard drive.
VBDRIVE_REMOT	&H1004	Remote network drive.
VBDRIVE_CDROM	&H1005	CD-ROM drive.
VBDRIVE_FLOPPY	&H1006	Floppy drive.
VBDRIVE_RAMDISK	&H1007	RAM drive
VBDRIVE_UNKNOWN	&H1010	Unknown device. This value should never be returned under Windows 3.1.
VBDRIVE_INVALID	&H0000	Invalid drive specified.

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbGetLongPtr

Declaration: Declare Function vbGetLongPtr Lib "VBASM.DLL" (*Variable* As Any) As Long

Description: Returns a long value that contains the address of *Variable*. The high-order word contains the segment portion and the low-order word contains the offset portion.

Note that the address returned for variable-length string variables is the address of Visual Basic's string header and not the address of the actual string text. To get the address of the actual string text, use vbSAdd and vbSSeg.

See Also: vbSAdd, vbSSeg, vbVarPtr, vbVarSeg

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbHiByte

Declaration: Declare Function vbHiByte Lib "VBASM.DLL" (ByVal *nValue* As Integer) As Integer

Description: Returns the high-order byte of the word specified by *nValue*.

See Also: [vbHiWord](#), [vbLoByte](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbHiWord

Declaration: Declare Function vbHiWord Lib "VBASM.DLL" (ByVal *nValue* As Long) As Integer

Description: Returns the high-order word of the long value specified by *nValue*.

See Also: [vbHiByte](#), [vbLoWord](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbInp

Declaration: Declare Function vbInp Lib "VBASM.DLL" (ByVal *nPort* As Integer) As Integer

Description: Reads a byte value from the I/O port specified by *nPort*.

Note that under Windows protected mode, some I/O ports may be in use by Windows and will not be available to your application.

See Also: [vbInpw](#), [vbOut](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbInpw

Declaration: Declare Function vbInpw Lib "VBASM.DLL" (ByVal *nPort* As Integer) As Integer

Description: Reads a word value from the I/O port specified by *nPort*.

Note that under Windows protected mode, some I/O ports may be in use by Windows and will not be available to your application.

See Also: [vbInp](#), [vbOutw](#)

VB-ASM, Version 1.30

Copyright © 1994-95 SoftCircuits Programming

vbInterrupt

[Example1](#)

[Example2](#)

Declaration: Declare Sub vbInterrupt Lib "VBASM.DLL" (ByVal *IntNum* As Integer, *InRegs* As VBREGS, *OutRegs* As VBREGS)

Description: Calls an interrupt (DS and ES are ignored). *IntNum* is the interrupt to be called. *InRegs* contains the registers to be passed to the interrupt, and *OutRegs* contains the registers returned by the interrupt.

```
Type VBREGS
AX As Integer      'General-purpose registers
BX As Integer
CX As Integer
DX As Integer
BP As Integer
SI As Integer
DI As Integer
Flags As Integer   'Flags register
DS As Integer      'Segment registers
ES As Integer
End Type
```

Using the **Flags** member, the following flags can be specified (note that on return, **Flags** contains all of the flags; however, only the following flags can be specified before the interrupt):

```
Global Const VBFLAGS_CARRY = &H1
Global Const VBFLAGS_PARITY = &H4
Global Const VBFLAGS_AUX = &H10
Global Const VBFLAGS_ZERO = &H40
Global Const VBFLAGS_SIGN = &H80
```

Note that under Windows protected mode, some DOS and BIOS interrupts that accept addresses will be expecting real-mode addresses and may behave unexpectedly when called from Windows.

See Also: [vbInterruptX](#), [vbRealModeIntX](#)

VB-ASM, Version 1.30

Copyright © 1994-95 SoftCircuits Programming

Copy

Close

An function to read the free space on a disk

```
Function GetDiskFreeSpace (iDisk) As Long
Dim inregs As VBREGS, outregs As VBREGS
Dim lDiskSpace&, lNumClusters&

    inregs.AX = &H3600
    inregs.DX = iDisk

    Call vbInterrupt(&H21, inregs, outregs)

    If outregs.AX = -1 Then
        MsgBox "Invalid drive specified", 48, "Invalid drive"
        lDiskSpace = 0&
    Else
        'AX = sectors/cluster
        'BX = Available clusters
        'CX = Bytes/sector
        lNumClusters = CLng(outregs.BX)
        'perform 2's complement if sign bit is set
        If lNumClusters < 0& Then lNumClusters = lNumClusters + 65536
        lDiskSpace = CLng(outregs.AX) * lNumClusters * CLng(outregs.CX)
    End If

    GetDiskFreeSpace = lDiskSpace

End Function
```


VB-ASM, Version 1.30

Copyright © 1994-95 SoftCircuits Programming

Copy

Close

An function to timestamp files

```
Function SetFileDateTime (iFileNum%, dFDate#, dFTime#) As Integer
Dim inregs As VBREGS, outregs As VBREGS
Dim j%, iDate%, iTime%
Dim sDate$, sTime$, sMsg$

If iFileNum <= 0 Then
    SetFileDateTime = -1
    Exit Function
End If

sDate = Format$(dFDate, "ddmmyy")
j = Val(Right$(sDate, 2))
If j < 80 Then j = j + 2000 Else j = j + 1900
j = j - 1980
iDate = vbShiftLeft(j, 9)
j = Val(Mid$(sDate, 3, 2)) And &HF
iDate = iDate Or vbShiftLeft(j, 5)
j = Val(Left$(sDate, 2)) And &H1F
inregs.DX = iDate Or j
'bits 0-4 = day
'bits 5-8 = month
'bits 9-15 = year relative to 1980

sTime = Format$(dFTime, "hhmm")
j = Val(Left$(sTime, 2))
iTime = vbShiftLeft(j, 11)
j = Val(Right$(sTime, 2)) And &H3F
inregs.CX = iTime Or vbShiftLeft(j, 5)
'bits 0-4 = seconds (not used here).
'As there are only 32 bits available, the value
'stored is half the actual value. E.g. if 20 is
'stored in bits 0-4 then the seconds value of the
'timestamp will be 40.
'bits 5-10 = minutes
'bits 11-15 = hours

inregs.AX = &H5701
inregs.BX = FileAttr(iFileNum, 2)
Call vbInterrupt(&H21, inregs, outregs)

If outregs.Flags And VBFLAGS_CARRY Then
    If outregs.AX = 1 Then
        sMsg = "Invalid function"
    ElseIf outregs.AX = 6 Then
        sMsg = "Invalid file handle"
    End If

    MsgBox sMsg, 48, "File Date/Time Error"
    SetFileDateTime = outregs.AX
Else
```

```
        SetFileDateTime = 0
    End If
End Function
```

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbInterruptX

Declaration: Declare Sub vbInterruptX Lib "VBASM.DLL" (ByVal *IntNum* As Integer, *InRegs* As VBREGS, *OutRegs* As VBREGS)

Description: Calls an interrupt (DS and ES are used). *IntNum* is the interrupt to be called. *InRegs* contains the registers to be passed to the interrupt, and *OutRegs* contains the registers returned by the interrupt.

See the [vbInterrupt](#) routine for additional information.

See Also: [vbInterrupt](#), [vbRealModeIntX](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbLoByte

Declaration: Declare Function vbLoByte Lib "VBASM.DLL" (ByVal *nValue* As Integer) As Integer

Description: Returns the low-order byte of the word specified by *nValue*.

See Also: [vbHiByte](#), [vbLoWord](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbLoWord

Declaration: Declare Function vbLoWord Lib "VBASM.DLL" (ByVal *nValue* As Long) As Integer

Description: Returns the low-order word of the long value specified by *nValue*.

See Also: [vbHiWord](#), [vbLoByte](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbMakeLong

Declaration: Declare Function vbMakeLong Lib "VBASM.DLL" (ByVal *nLoWord* As Integer, ByVal *nHiWord* As Integer) As Long

Description: Combines two word values into a long integer value.

See Also: [vbMakeWord](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbMakeWord

Declaration: Declare Function vbMakeWord Lib "VBASM.DLL" (ByVal *nLoByte* As Integer, ByVal *nHiByte* As Integer) As Integer

Description: Combines two byte values into a word value.

See Also: [vbMakeLong](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbOut

Declaration: Declare Sub vbOut Lib "VBASM.DLL" (ByVal *nPort* As Integer, ByVal *nData* As Integer)

Description: Sends a byte value to the I/O port specified by *nData*.

Note that under Windows protected mode, some I/O ports may be in use by Windows and will not be available to your application.

See Also: [vbInp](#), [vbOutw](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbOutw

Declaration: Declare Sub vbOutw Lib "VBASM.DLL" (ByVal *nPort* As Integer, ByVal *nData* As Integer)

Description: Sends a word value to the I/O port specified by *nData*.

Note that under Windows protected mode, some I/O ports may be in use by Windows and will not be available to your application.

See Also: [vbInpw](#), [vbOut](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbPeek

Declaration: Declare Function vbPeek Lib "VBASM.DLL" (ByVal *nSegment* As Integer, ByVal *nOffset* As Integer) As Integer

Description: Returns the byte value at the memory location specified by *nSegment* and *nOffset*.

Due to the nature of Windows protected mode, *nSegment* must be &H0000, &H0040, &HA000, &HB000, &HC000, &HD000, &HE000 or &HF000 or the call is ignored.

See Also: [vbPeekw](#), [vbPoke](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbPeekw

Declaration: Declare Function vbPeekw Lib "VBASM.DLL" (ByVal *nSegment* As Integer, ByVal *nOffset* As Integer) As Integer

Description: Returns the word value at the memory location specified by *nSegment* and *nOffset*.

Due to the nature of Windows protected mode, *nSegment* must be &H0000, &H0040, &HA000, &HB000, &HC000, &HD000, &HE000 or &HF000 or the call is ignored.

See Also: [vbPeek](#), [vbPokew](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbPoke

Declaration: Declare Sub vbPoke Lib "VBASM.DLL" (ByVal *nSegment* As Integer, ByVal *nOffset* As Integer, ByVal *nValue* As Integer)

Description: Writes the byte value specified by *nValue* to the memory location specified by *nSegment* and *nOffset*.

Due to the nature of Windows protected mode, *nSegment* must be &H0000, &H0040, &HA000, &HB000, &HC000, &HD000, &HE000 or &HF000 or the call is ignored.

See Also: [vbPeek](#), [vbPokew](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbPokew

Declaration: Declare Sub vbPokew Lib "VBASM.DLL" (ByVal *nSegment* As Integer, ByVal *nOffset* As Integer, ByVal *nValue* As Integer)

Description: Writes the word value specified by *nValue* to the memory location specified by *nSegment* and *nOffset*.

Due to the nature of Windows protected mode, *nSegment* must be &H0000, &H0040, &HA000, &HB000, &HC000, &HD000, &HE000 or &HF000 or the call is ignored.

See Also: [vbPeekw](#), [vbPoke](#)

VB-ASM, Version 1.30

Copyright © 1994-95 SoftCircuits Programming

vbRealModelntX

[Example1](#)

[Example2](#)

Declaration: Declare Function vbRealModelntX Lib "VBASM.DLL" (ByVal *IntNum* As Integer, *InRegs* As VBREGS, *OutRegs* As VBREGS) As Integer

Description: Calls an interrupt (DS and ES are used). *IntNum* is the interrupt to be called. *InRegs* contains the registers to be passed to the interrupt, and *OutRegs* contains the registers returned by the interrupt. vbRealModelntX returns True if successful, False otherwise. Note that a return value of True indicates that this function was successful and not that the interrupt service being invoked was successful.

This function is similar to the [vbInterruptX](#) procedure except that vbRealModelntX switches the processor to real-mode before invoking the interrupt. Normally, you will want to use [vbInterrupt](#) or [vbInterruptX](#). These two functions invoke interrupts under Windows protected mode. Depending on the interrupt service being invoked, Windows may automatically perform the equivalent of vbRealModelntX by switching to real mode and re-issuing the interrupt. In other cases, Windows will even service the interrupt itself.

So, you might ask, if Windows either services the interrupt or passes it along to the real mode interrupt handler, why would it ever be necessary to use vbRealModelntX to invoke a real mode handler directly? Let's say you want to invoke a DOS service that fills a buffer with information. Many such services require you to pass the address of the buffer in registers. The problem is that the address of a buffer within your Windows program will be a protected mode address. Moreover, the physical location of that buffer will most likely be outside of the 1MB memory area available to code running in real mode (i.e., a real mode interrupt handler). Clearly, if Windows passes such an interrupt request to the real mode handler, there is very little chance that the data provided by the interrupt would ever make it to your program's buffer.

Now in some cases, Windows will automatically allocate memory in real mode, copy your buffer to this location, change the registers to point to the new buffer, invoke the real mode handler and, finally, copy the results back to your buffer. Unfortunately, there are many services for which Windows does *not* do this. For example, let's invoke the DOS [TrueName](#) service (interrupt &H21, function &H60) which takes a partial path or filename and returns a normalized, fully qualified filename. This is an undocumented service and so, not surprisingly, Windows provides no behind-the-scenes translation for us. We must allocate real mode memory for the buffers and pass real mode addresses in the registers. Fortunately, the Windows API functions GlobalDOSAlloc and GlobalDOSFree allow us to do just that. But now that we are passing real mode addresses, we must ensure that the service is never handled by a protected mode handler. Since a Windows driver could be installed to service this interrupt, or maybe future versions of Windows will support it, we need to use vbRealModelntX so that we know it will always be sent to the real mode interrupt handler.

See the [vbInterrupt](#) routine for additional information.

See Also: [vbInterrupt](#), [vbInterruptX](#)

VB-ASM, Version 1.30

Copyright © 1994-95 SoftCircuits Programming

Copy

Close

A TrueName function

The following code shows how we could implement a TrueName function. Note that this code is provided as an example only. Since the service is not documented, it may very well not be supported in future versions of DOS or Windows. Also note that statements that are too long to fit on a single line are joined on separate lines by an underscore (_). In VB version 3.0 and earlier, you must delete the underscore and combine the two lines.

```
'Windows API declarations
Declare Function GlobalDOSAlloc& Lib "Kernel" (ByVal cbAlloc&)
Declare Function GlobalDOSFree% Lib "Kernel" (ByVal uSelector%)

Function TrueName (PartialPath As String) As String
    Dim FileNamePtr As Long, FullPathPtr As Long
    Dim i As Long, buffer As String, myRegs As VBREGS

    'Allocate input and output buffers in real-mode memory
    FileNamePtr = GlobalDOSAlloc(128)
    If FileNamePtr = 0 Then Exit Function
    FullPathPtr = GlobalDOSAlloc(128)
    If FullPathPtr = 0 Then
        i = GlobalDOSFree(vbLoWord(FileNamePtr))
        Exit Function
    End If

    'Copy Chr$(0)-terminated partial path to real-mode buffer
    Call vbSetData(vbMakeLong(0, vbLoWord(FileNamePtr)), _
        ByVal PartialPath, Len(PartialPath) + 1)

    'Set up interrupt registers
    myRegs.AX = &H6000
    myRegs.DS = vbHiWord(FileNamePtr)
    myRegs.SI = 0
    myRegs.ES = vbHiWord(FullPathPtr)
    myRegs.DI = 0

    'Call DOS using real-mode interrupt
    If vbRealModeIntX(&H21, myRegs, myRegs) Then
        If (myRegs.Flags And VBFLAGS_CARRY) = 0 Then
            'Allocate room for the result
            buffer = Space$(128)
            'Copy result to buffer
            Call vbGetData(vbMakeLong(0, vbLoWord(FullPathPtr)), _
                ByVal buffer, Len(buffer))
            'Set return value
            TrueName = Left$(buffer, InStr(buffer, Chr$(0)) - 1)
        End If
    End If

    'Free allocated memory
    i = GlobalDOSFree(vbLoWord(FileNamePtr))
    i = GlobalDOSFree(vbLoWord(FullPathPtr))
```

End Function

VB-ASM, Version 1.30

Copyright © 1994-95 SoftCircuits Programming

Copy

Close

Get drive serial number

Yet another example of using vbRealModeX is the following routine which returns the serial number for the specified drive. Again, this routine makes use of undocumented DOS services which may not be supported by future versions of DOS or Windows.

```
Declare Function GlobalDOSAlloc& Lib "Kernel" (ByVal cbAlloc&)
Declare Function GlobalDOSFree% Lib "Kernel" (ByVal uSelector%)

Type DiskInfo
    infoLevel As Integer
    serialNum As Long
    volLabel As String * 11
    fSysType As String * 8
End Type

Function GetSerialNum (drive As String) As String
    Dim myRegs As VBREGS, di As DiskInfo, buffAddr&, i%

    'Since this is an undocumented service, we cannot count on
    'Windows supporting it under Windows protected mode so we
    'must allocate real mode memory and call the real mode handler

    'Allocate real mode memory
    buffAddr& = GlobalDOSAlloc(Len(di))
    'Abort if error
    If buffAddr& = 0 Then Exit Function

    'Invoke get/set disk serial number service using real mode int
    'Note: requires DOS 4 or higher and is an undocumented service
    myRegs.AX = &H6900
    myRegs.BX = (Asc(UCase$(drive)) - Asc("A")) + 1
    myRegs.DX = 0
    myRegs.DS = vbHiWord(buffAddr&)

    'Invoke real mode handler
    i = vbRealModeIntX(&H21, myRegs, myRegs)

    If myRegs.Flags And VBFLAGS_CARRY Then
        'Unsuccessful if carry flag set
        GetSerialNum = ""
    Else
        'Copy result to local structure
        Call vbGetData(vbMakeLong(0, vbLoWord(buffAddr&)), _
            di, Len(di))
        'Form and return result
        GetSerialNum = Hex$(vbHiWord(di.serialNum)) & "-" & _
            Hex$(vbLoWord(di.serialNum))
    End If

    'Free allocated memory
    i = GlobalDOSFree(vbLoWord(buffAddr&))
```

End Function

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbRecreateCtrl

Declaration: Declare Function vbRecreateCtrl Lib "VBASM.DLL" (*Ctrl* As Any) As Integer

Description: This function destroys the Visual Basic control specified by *Ctrl* and then recreates it. This is useful when you need to specify attributes, for the window associated with a control, that cannot normally be modified once the window has been created.

If successful, this function returns True. False is returned if the control could not be destroyed and recreated. Note: This function should not be used for container controls.

In the process of destroying and recreating the control, this function saves and restores both font information and standard properties stored only in the window and which would otherwise be erased when the window is destroyed (this includes the Enabled, TabIndex, TabStop, and Visible properties).

One use for this function is to set window style bits. Windows stores a number of style bits that specify the appearance and behavior of the window associated with each form and control. You should use the SetWindowWord() API function to change those style bits that cannot be modified through properties. However, there are several bits that are ignored if they are set after the window has been created. These bits must instead be specified when the window is created. Visual Basic handles creating all windows automatically and does not allow you to specify style bits used to create the window. The vbRecreateCtrl function allows you to change the style bits associated with a form or control and then destroy the window and recreate it.

See Also: [vbGetCtrlModel](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbSAdd

Declaration: Declare Function vbSAdd Lib "VBASM.DLL" (*Variable* As String) As Integer

Description: Returns the offset address of the string text of a variable-length string variable.

See Also: [vbSSeg](#), [vbVarPtr](#), [vbGetLongPtr](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbSetData

Declaration: Declare Sub vbSetData Lib "VBASM.DLL" (ByVal *Pointer* As Long, *Variable* As Any,
 ByVal *nCount* As Integer)

Description: Copies data from *Variable* to the memory location pointed to by *Pointer*. *nCount* specifies
 the number of bytes to be copied. This function is useful if you need access to data not
 within your program.

See Also: [vbGetData](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbShiftLeft

Declaration: Declare Function vbShiftLeft Lib "VBASM.DLL" (ByVal *nValue* As Integer, ByVal *nBits* As Integer) As Integer

Description: Shifts the bits of *nValue* to the left. *nBits* specifies how many bit positions each bit is shifted. For example, vbShiftLeft(&H1,1) returns &H2, vbShiftLeft(&H1,4) returns &H10, etc.

See Also: [vbShiftLeftLong](#), [vbShiftRight](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbShiftLeftLong

Declaration: Declare Function vbShiftLeftLong Lib "VBASM.DLL" (ByVal *nValue* As Long, ByVal *nBits* As Integer) As Long

Description: Works the same as vbShiftLeft except that vbShiftLeftLong works with variables of type Long. See [vbShiftLeft](#) for more information.

See Also: [vbShiftLeft](#), [vbShiftRightLong](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbShiftRight

Declaration: Declare Function vbShiftRight Lib "VBASM.DLL" (ByVal *nValue* As Integer, ByVal *nBits* As Integer) As Integer

Description: Shifts the bits of *nValue* to the right. *nBits* specifies how many bit positions each bit is shifted. For example, vbShiftRight(&H10,1) returns &H8, vbShiftRight(&H10,4) returns &H1, etc.

See Also: [vbShiftLeft](#), [vbShiftRightLong](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbShiftRightLong

Declaration: Declare Function vbShiftRightLong Lib "VBASM.DLL" (ByVal *nValue* As Long, ByVal *nBits* As Integer) As Long

Description: Works the same as vbShiftRight except that vbShiftRightLong works with variables of type Long. See [vbShiftRight](#) for more information.

See Also: [vbShiftLeftLong](#), [vbShiftRight](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbSSeg

Declaration: Declare Function vbSSeg Lib "VBASM.DLL" (*Variable* As String) As Integer

Description: Returns the segment address of the string text of a variable-length string variable.

See Also: [vbSAdd](#), [vbVarSeg](#), [vbGetLongPtr](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbVarPtr

Declaration: Declare Function vbVarPtr Lib "VBASM.DLL" (*Variable As Any*) As Integer

Description: Returns the offset portion of Variable's address.

Note that the address returned for variable-length string variables is the address of Visual Basic's string header and not the address of the actual string text. To get the address of the actual string text, use [vbSAdd](#) and [vbSSeg](#).

See Also: [vbSAdd](#), [vbVarSeg](#), [vbGetLongPtr](#)

VB-ASM, Version 1.30
Copyright © 1994-95 SoftCircuits Programming

vbVarSeg

Declaration: Declare Function vbVarSeg Lib "VBASM.DLL" (*Variable* As Any) As Integer

Description: Returns the segment portion of Variable's address.

Note that the address returned for variable-length string variables is the address of Visual Basic's string header and not the address of the actual string text. To get the address of the actual string text, use [vbSAdd](#) and [vbSSeg](#).

See Also: [vbSSeg](#), [vbVarPtr](#), [vbGetLongPtr](#)

VB-ASM, Version 1.30

Copyright © 1994-95 SoftCircuits Programming

Revision History

This section documents the changes and additions made to VB-ASM. Revisions are listed with the most recent version first.

Version:	Modification(s):
1.30	<ul style="list-style-type: none">+ Added new vbShiftLeftLong and vbShiftRightLong functions.+ Corrected problem in Snooper demo that caused the disk space routines to fail on drives as large as 1GB.
1.20	<ul style="list-style-type: none">+ Added new vbGetCtrlName and vbGetCtrlHwnd functions.+ The DLL now contains an internal version information resource. This information is used by VER.DLL during installation to determine if the version being installed is newer or older than any existing version.+ After receiving a number of questions about how to change the alignment of a multi-line text box at run-time, the LBS_HGHT demo was replaced by the TXTALIGN demo.+ Changed DLL to use small memory model making the DLL file substantially smaller.+ Minor changes to VB-ASM documentation (this file).
1.10	<ul style="list-style-type: none">+ Added new vbGetDriveType function.+ Due to some assembler directives being disabled in order to work correctly with Windows, some routines did not correctly restore the SI and DI registers. In addition, this caused a bug in vbRealModeX where variables were not allocated correctly. All known problems have been corrected.+ Fixed bug where vbInterrupt and vbInterruptX did not return the correct value for OutRegs.ES.+ Optimizations made to VBASM.DLL.+ Changed the declarations for vbGetCtrlModel and vbRecreateCtrl to accept control or form arguments directly. Previously, an argument of type long was required.+ Added VB prefix to the VBREGS data type and flag constants for consistency with other VB-ASM declarations and to help minimize conflict with Windows API declarations.+ Many modifications to Snooper demo program.+ Minor changes to VB-ASM documentation (this file).
1.00	<ul style="list-style-type: none">+ Original version.

