

Visual Basic Add-on DLL Version 1.02

This Dynamic Link Library (DLL) provides the following additional functions for Visual Basic

Huge Array Support - Support for arrays which exceed Visual Basic's limitations

Disk Information Support - Access to disk/diskette information not available through Visual Basic

THE INFORMATION PROVIDED IN THIS SOFTWARE AND ANY DOCUMENTATION MAY ACCOMPANY THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. THE USER ASSUMES THE ENTIRE RISK AS TO THE ACCURACY AND THE USE OF THIS SOFTWARE.

This software may be copied and distributed subject to the following conditions:

- 1) All documentation must be copied without modification and all pages must be included;
- 2) All files on the disk(s) must be copied without modification;
- 3) All components must be distributed together; and
- 4) This software may not be distributed for profit.

Copyright 1991 by:

Michael A. Stewart
539 Dutch Neck Road
East Windsor, NJ 08520

Visual Basic is a trademark of Microsoft Corporation

If you have any questions about its functions or suggestions for its future enhancement the author can be contacted through the following services:

CompuServe ID: 76234,3314
Prodigy ID: HWKS33A
AOL ID: Michae1334
ILINK Windows, Basic, and Win.App.Dev Conferences.

Huge Array Support

The huge array functions provide the ability to utilize arrays which exceed the Visual Basic limitations of 32,767 elements and/or 65,536 bytes total size, and to redimension arrays without erasing their contents.

To use the huge array functions, copy the appropriate declarations contained in the VBADDONS.TXT file to the global module of your program, and ensure that the VBADDONS.DLL is in the normal Windows program search path.

Declare Statements - Declarations required to use the functions in Visual Basic.

Error Codes - Descriptions of possible huge array error codes.

Functions - Descriptions of the various huge array functions provided by the DLL.

Examples for the use of all huge array functions may be found in the HUGESUPP.BAS file.

Huge Arrays - Declare Statements

To use the huge array functions the following declare statements must be placed in the global module of your Visual Basic program:

```
Declare Function HugeCurrency Lib "vbaddons.dll"  
    (ByVal hArray%, ByVal el&) As Currency  
Declare Function HugeDim Lib "vbaddons.dll"  
    (ByVal recsize%, ByVal limit&) As Integer  
Declare Function HugeDouble Lib "vbaddons.dll"  
    (ByVal hArray%, ByVal el&) As Double  
Declare Function HugeErase Lib "vbaddons.dll"  
    (ByVal hArray%) As Integer  
Declare Function HugeGetElement Lib "vbaddons.dll"  
    (ByVal Index%, ByVal element&, buffer As Any) As Integer  
Declare Function HugeInt Lib "vbaddons.dll"  
    (ByVal hArray%, ByVal el&) As Integer  
Declare Function HugeLong Lib "vbaddons.dll"  
    (ByVal hArray%, ByVal el&) As Long  
Declare Function HugeNumArrays Lib "vbaddons.dll"  
    () As Integer  
Declare Function HugeRedim Lib "vbaddons.dll"  
    (ByVal hArray%, ByVal limit&) As Integer  
Declare Function HugeSetElement Lib "vbaddons.dll"  
    (ByVal Index%, ByVal element&, buffer As Any) As Integer  
Declare Function HugeSingle Lib "vbaddons.dll"  
    (ByVal hArray%, ByVal el&) As Single  
Declare Function HugeUbound Lib "vbaddons.dll"  
    (ByVal hArray%) As Integer
```

These declare statements are provided in the VBADDONS.TXT file.

Huge Array Error Codes

The following error codes may be returned by the various Huge Array functions:

Error Code	Description
0	No error has occurred
-1	Insufficient memory available to create the array.
-2	The number of available huge arrays has been exhausted.
-3	The requested element size is invalid. (see HugeDim for details)
-4	Invalid element number (must be in the range of 0 to the upper boundary).
-5	Invalid array handle

Huge Array Functions

The following functions are provided:

HugeCurrency - Retrieve Currency value from a Huge Array
HugeDim - Dimension a Huge Array
HugeDouble - Retrieve Double value from a Huge Array
HugeErase - Erase a Huge Array
HugeGetElement - Retrieve an element from a Huge Array
HugeInt - Retrieve Integer value from a Huge Array
HugeLong - Retrieve Long value from a Huge Array
HugeNumArrays - Retrieve the number of available Huge Arrays
HugeRedim - Redimension a Huge Array
HugeSetElement - Store an element in a Huge Array
HugeSingle - Retrieve Single value from a Huge Array
HugeUbound - Retrieve Upper Boundary of a Huge Array
Huge Array Error Codes - Error return codes for huge arrays

HugeCurrency

Action:

Retrieves a currency value from a huge array.

Syntax:

HugeCurrency(*hArray As Integer*, *ElementNumber As Long*) As Currency

Remarks:

The **HugeCurrency** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by HugeDim or HugeRedim .
<i>ElementNumber</i>	The number of the element within the huge array. This number must be in the range of 0 to the upper boundary of the huge array.

Notes:

This function does not return an error code or perform any data type checking, therefore it should only be used when you are absolutely sure that the values of *hArray* and *ElementNumber* are valid and that the array contains currency values.

HugeDim Statement

Action:

Dimensions a huge array and returns a handle to that array.

Syntax:

HugeDim(*ElementSize* **As Integer**, *UpperBound* **As Long**) **As Integer**

Remarks:

The **HugeDim** statement uses the following arguments:

Argument	Description
<i>ElementSize</i>	The size of each element in the array (2 for Integer, 4 for single, etc). The Len() function may be used to determine this value.
<i>UpperBound</i>	The upper boundary of the array. The lower boundary of all huge arrays is 0.

The **HugeDim** statement returns the following values:

Returns	Description
Integer	If no error has occurred, the array handle for use by other Huge function is returned. If an error has occurred, a negative value representing the <u>error code</u> is returned.

Notes:

The Windows API - GlobalAlloc function is utilized and the total size of any array is limited to 1MB in Standard Mode and 64MB in Enhanced Mode.

You cannot create a huge array of variable length strings, or of a user defined type which contains variable length strings. Only fixed length strings may be used in huge arrays.

If the total size (in bytes) of an array is going to be greater than 64K the size of each element has to be an integer power of 2 (1, 2, 4, 8, and so forth).

HugeDouble

Action:

Retrieves a double precision value from a huge array.

Syntax:

HugeDouble(*hArray* As Integer, *ElementNumber* As Long) As Double

Remarks:

The **HugeDouble** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by HugeDim or HugeRedim .
<i>ElementNumber</i>	The number of the element within the huge array. This number must be in the range of 0 to the upper boundary of the huge array.

Notes:

This function does not return an error code or perform any data type checking, therefore it should only be used when you are absolutely sure that the values of *hArray* and *ElementNumber* are valid and that the array contains double precision values.

HugeErase

Action:

Erases a previously dimensioned huge array.

Syntax:

HugeErase(*hArray* As Integer) As Integer

Remarks:

The **HugeErase** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by HugeDim or HugeRedim .

The **HugeErase** statement returns the following values:

Returns	Description
Integer	If the <i>hArray</i> is valid a 0 is returned. If <i>hArray</i> is not valid, a negative value representing the error code is returned.

Notes:

HugeErase must be used for all array created by [HugeDim](#) or [HugeRedim](#). Failure to do so will cause the number of available huge arrays to become exhausted.

HugeGetElement

Action: Retrieves an element from a huge array.

Syntax:

HugeGetElement(*hArray* As Integer, *ElementNumber* As Long, *Buffer* as Any) As Integer

Remarks:

The **HugeGetElement** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by <u>HugeDim</u> or <u>HugeRedim</u> .
<i>ElementNumber</i>	The number of the element within the huge array. This number must be in the range of 0 to the upper boundary of the huge array.
<i>Buffer</i>	The variable to receive the contents of the huge array element.

The **HugeGetElement** statement returns the following values:

Returns	Description
Integer	If the <i>Variable</i> has been retrieved from the array a 0 is returned. If an error has occurred, a negative value representing the <u>error code</u> is returned.

Notes:

This function does perform any data type checking, therefore the length of *Variable* must exactly match the *ElementSize* specified in the HugeDim statement for the array.

HugeInt

Action:

Retrieves an integer value from a huge array.

Syntax:

HugeInt(*hArray* As Integer, *ElementNumber* As Long) As Integer

Remarks:

The **HugeInt** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by HugeDim or HugeRedim .
<i>ElementNumber</i>	The number of the element within the huge array. This number must be in the range of 0 to the upper boundary of the huge array.

Notes:

This function does not return an error code or perform any data type checking, therefore it should only be used when you are absolutely sure that the values of *hArray* and *ElementNumber* are valid and that the array contains integer values.

HugeLong

Action:

Retrieves a long value from a huge array.

Syntax:

HugeLong(*hArray* As Integer, *ElementNumber* As Long) As Long

Remarks:

The **HugeLong** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by HugeDim or HugeRedim .
<i>ElementNumber</i>	The number of the element within the huge array. This number must be in the range of 0 to the upper boundary of the huge array.

Notes:

This function does not return an error code or perform any data type checking, therefore it should only be used when you are absolutely sure that the values of *hArray* and *ElementNumber* are valid and that the array contains long values.

HugeNumArrays

Action:

Returns the number of available huge arrays.

Syntax:

HugeNumArrays As Integer

Remarks:

The **HugeNumArrays** statement returns the following values:

Returns	Description
Integer	The number of huge arrays currently available.

HugeRedim

Action:

Redimensions a previously dimensioned huge array.

Syntax:

HugeRedim(*hArray As Integer*, *UpperBound As Long*) **As Integer**

Remarks:

The **HugeRedim** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by HugeDim or HugeRedim .
<i>UpperBound</i>	The new upper boundary of the array. The lower boundary of all huge arrays is 0.

The **HugeErase** statement returns the following values:

Returns	Description
Integer	If the array has been redimensioned a 0 is returned. If the array was not redimensioned, a negative value representing the error code is returned.

Notes:

Unlike Visual Basic's ReDim, **HugeRedim** preserves the previous contents of the array. [HugeErase](#) followed by [HugeDim](#) should be used to erase the contents of the array.

HugeSetElement

Action:

Stores an element in a huge array.

Syntax:

HugeSetElement(*hArray* As Integer, *ElementNumber* As Long, *Buffer* as Any) As Integer

Remarks:

The **HugeSetElement** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by <u>HugeDim</u> or <u>HugeRedim</u> .
<i>ElementNumber</i>	The number of the element within the huge array. This number must be in the range of 0 to the upper boundary of the huge array.
<i>Buffer</i>	The variable containing the data to be stored in the huge array element.

The **HugeSetElement** statement returns the following values:

Returns	Description
Integer	If the <i>Variable</i> has been stored in the array a 0 is returned. If an error has occurred, a negative value representing the <u>error code</u> is returned.

Notes:

This function does perform any data type checking, therefore the length of *Variable* must exactly match the *ElementSize* specified in the HugeDim statement for the array.

HugeSingle

Action:

Retrieves a single precision value from a huge array.

Syntax:

HugeSingle(*hArray* As Integer, *ElementNumber* As Long) As Single

Remarks:

The **HugeSingle** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by HugeDim or HugeRedim .
<i>ElementNumber</i>	The number of the element within the huge array. This number must be in the range of 0 to the upper boundary of the huge array.

Notes:

This function does not return an error code or perform any data type checking, therefore it should only be used when you are absolutely sure that the values of *hArray* and *ElementNumber* are valid and that the array contains single precision values.

HugeUbound

Action:

Returns the upper boundary of a previously dimensioned huge array.

Syntax:

HugeUbound(*hArray* As Integer) As Integer

Remarks:

The **HugeUbound** statement uses the following arguments:

Argument	Description
<i>hArray</i>	The array handle as returned by HugeDim or HugeRedim .

The **HugeErase** statement returns the following values:

Returns	Description
Integer	If the <i>hArray</i> is valid the upper boundary of the huge array is returned. If <i>hArray</i> is not valid, a negative value representing the error code is returned.

Disk Information Support

The Disk Information Support provides functions to access and alter disk information.

To use the Disk Information Functions, copy the appropriate declarations contained in the VBADDONS.TXT file to the global module of your program, and ensure that the VBADDONS.DLL is in the normal Windows program search path.

Data Type Statements - Data Types required to use the functions in Visual Basic.

Declare Statements - Declarations required to use the functions in Visual Basic.

Functions - Descriptions of the Disk Information functions provided by the DLL

Examples for the use of the disk support functions may be found in the DISKSUPP.BAS file.

Disk Information - Data Types

The following data types must be placed in the global module of the program:

```
Type FindDataType
    reserved As String * 21
    FileAttr As String * 1
    FileTime As Integer
    FileDate As Integer
    FileSize As Long
    Filename As String * 13
End Type
```

FileTime is returned as:

Bits	Contents
0-4	Number of 2-second increments (0-29)
5-10	Minutes (0-59)
11-15	Hours(0-23)

File Date is returned as:

Bits	Contents
0-4	Day of month (1-31)
5-8	Month (1-12)
9-15	Year (relative to 1980)

These statements may be found in the VBADDONS.TXT file.

The following constants should be placed in the global module of the program:

```
'
'   Bitmasks for FileAttr field of FindDataType
'
Global Const FILE_NORMAL = 0   ' Normal files
Global Const FILE_RDONLY = 1   ' Read only file
Global Const FILE_HIDDEN = 2   ' Hidden file
Global Const FILE_SYSTEM = 4   ' System file
Global Const FILE_VOLID = 8    ' Volume ID file
Global Const FILE_SUBDIR = 16  ' Subdirectory
Global Const FILE_ARCH = 32    ' Archive flag
```

These statements may be found in the VBADDONS.DSK file.

Disk Information - Declare Statements

To use the Disk Information functions the following declare statements must be placed in the global module of your Visual Basic program:

```
Declare Function DiskGetFreeSpace Lib "vbaddons.dll"  
    (ByVal DriveLetter As String) As Long  
Declare Function DiskGetFirstFile Lib "vbaddons.dll"  
    (ByVal StartString As String, ByVal AttrFlags As Integer,  
    FindData As FindDataType) As Integer  
Declare Function DiskGetNextFile Lib "vbaddons.dll"  
    (FindData As FindDataType) As Integer  
Declare Function DiskSetLabel Lib "vbaddons.dll"  
    (ByVal DriveLetter As String, ByVal NewLabel As String)  
    As Integer
```

These statements are provided in the VBADDONS.TXT file.

Disk Support Functions

The following functions are provided:

DiskGetFreeSpace - Returns the amount of free space on a disk

DiskGetFirstFileDisk and DiskGetNextFile - Return file information from a directory

DiskSetLabel - Sets the disk label

DiskGetFreeSpace Statement

Action:

Returns the amount of free space on a disk.

Syntax:

DiskGetFreeSpace(ByVal *DriveLetter* As String) As Long

Remarks:

The **DiskGetFreeSpace** statement uses the following arguments:

Argument	Description
<i>DriveLetter</i>	A string whose first character contains the desired drive "A", "B", etc.

The **DiskGetFreeSpace** statement returns the following values:

Returns	Description
Long	If no error has occurred, the amount of free space on the disk. If an error has occurred -1 is returned.

DiskGetFirstFile Statement

DiskGetNextFile Statement

Action:

Uses in combination with each other to read the contents of a disk directory.

Syntax:

```
DiskGetFirstFile(ByVal StartString As String, ByVal AttrFlags As Integer,  
    FindData As FindDataType) As Integer  
DiskGetNextFile(FindData As FindDataType) As Integer
```

Remarks:

The **DiskGetFirstFile** and **DiskGetNextFile** statements use the following arguments:

Argument	Description														
<i>StartString</i>	<p>A string which contains a drive:pathname.filename.ext string to begin the search. The drive letter is optional if the current drive is to be used. The pathname is optional if the current path is to be used. Standard DOS wildcards may be used in the filename and ext sections of the string.</p> <p>Examples:</p> <p>"b:*.*)"</p> <p>"c:\windows*.ini"</p>														
<i>AttrFlags</i>	<p>An integer value which specifies which file types are to be included in the search as follows:</p> <table><tr><td>0</td><td>Normal files - No read/write restrictions</td></tr><tr><td>1</td><td>Read only files and normal files</td></tr><tr><td>2</td><td>Hidden files and normal files</td></tr><tr><td>4</td><td>System files and normal files</td></tr><tr><td>8</td><td>Volume ID only</td></tr><tr><td>16</td><td>Subdirectories and normal files</td></tr><tr><td>32</td><td>Archive files only</td></tr></table>	0	Normal files - No read/write restrictions	1	Read only files and normal files	2	Hidden files and normal files	4	System files and normal files	8	Volume ID only	16	Subdirectories and normal files	32	Archive files only
0	Normal files - No read/write restrictions														
1	Read only files and normal files														
2	Hidden files and normal files														
4	System files and normal files														
8	Volume ID only														
16	Subdirectories and normal files														
32	Archive files only														
<i>FindData</i>	<p>A user data type (FindDataType) which will receive the information for a file which meets the search criteria. for DiskGetNextFile this must be the same variable used for the GetFirstFile.</p>														

The **DiskGetFirstFile** and **DiskGetNextFile** statements return following values:

Returns	Description
Integer	If a matching file was found a 1 is returned. If no matching file was found a 0 is returned.

Notes:

Correct usage requires that a **DiskGetFirstFile** be utilized to begin the search and **DiskGetNextFile** used until a return value of 0 is received.

DiskSetLabel Statement

Action:

Sets the disk label to a specified string value.

Syntax:

DiskSetLabel(ByVal *DriveLetter* As String, ByVal *NewLabel* As String) As Integer

Remarks:

The **DiskSetLabel** statement uses the following arguments:

Argument	Description
<i>DriveLetter</i>	A string whose first character contains the desired drive "A", "B", etc.
<i>NewLabel</i>	A string containing 1 to 11 characters which will be used as the new disk label.

The **DiskSetLabel** statement returns the following values:

Returns	Description
Integer	If no error has occurred, a 0 is returned. If an error has occurred, a non-zero is returned.